B 9

```
NNN         NNN  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT     AAAAAAAAA        CCCCCCCCCCCC  PPPPPPPPPPPP
NNN         NNN  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT     AAAAAAAAA        CCCCCCCCCCCC  PPPPPPPPPPPP
NNN         NNN  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT     AAAAAAAAA        CCCCCCCCCCCC  PPPPPPPPPPPP
NNN         NNN  EEE                    TTT        AAA         AAA  CCC              PPP         PPP
NNN         NNN  EEE                    TTT        AAA         AAA  CCC              PPP         PPP
NNN         NNN  EEE                    TTT        AAA         AAA  CCC              PPP         PPP
NNNNNN      NNN  EEE                    TTT        AAA         AAA  CCC              PPP         PPP
NNNNNN      NNN  EEE                    TTT        AAA         AAA  CCC              PPP         PPP
NNNNNN      NNN  EEE                    TTT        AAA         AAA  CCC              PPP         PPP
NNN   NNN   NNN  EEEEEEEEEEEE           TTT        AAA         AAA  CCC              PPPPPPPPPPPP
NNN   NNN   NNN  EEEEEEEEEEEE           TTT        AAA         AAA  CCC              PPPPPPPPPPPP
NNN   NNN   NNN  EEEEEEEEEEEE           TTT        AAA         AAA  CCC              PPPPPPPPPPPP
NNN      NNNNNN  EEE                    TTT        AAAAAAAAAAAAAAA  CCC              PPP
NNN      NNNNNN  EEE                    TTT        AAAAAAAAAAAAAAA  CCC              PPP
NNN      NNNNNN  EEE                    TTT        AAAAAAAAAAAAAAA  CCC              PPP
NNN         NNN  EEE                    TTT        AAA         AAA  CCC              PPP
NNN         NNN  EEE                    TTT        AAA         AAA  CCC              PPP
NNN         NNN  EEE                    TTT        AAA         AAA  CCC              PPP
NNN         NNN  EEEEEEEEEEEEEEE        TTT        AAA         AAA     CCCCCCCCCCCC  PPP
NNN         NNN  EEEEEEEEEEEEEEE        TTT        AAA         AAA     CCCCCCCCCCCC  PPP
NNN         NNN  EEEEEEEEEEEEEEE        TTT        AAA         AAA     CCCCCCCCCCCC  PPP
```

```
_S

Ps
--
NE

NE

NE

NE

$R
```

```
**FILE**ID**NETDRVSES                           L 11




NN      NN  EEEEEEEEEE  TTTTTTTTTT  DDDDDDDD    RRRRRRRR    VV      VV    SSSSSSSS  EEEEEEEEEE    SSSSSSSS
NN      NN  EEEEEEEEEE  TTTTTTTTTT  DDDDDDDD    RRRRRRRR    VV      VV    SSSSSSSS  EEEEEEEEEE    SSSSSSSS
NN      NN  EE              TT      DD      DD  RR      RR  VV      VV  SS          EE            SS
NN      NN  EE              TT      DD      DD  RR      RR  VV      VV  SS          EE            SS
NNNN    NN  EE              TT      DD      DD  RR      RR  VV      VV  SS          EE            SS
NNNN    NN  EE              TT      DD      DD  RR      RR  VV      VV  SS          EE            SS
NN  NN  NN  EEEEEEEE        TT      DD      DD  RRRRRRRR    VV      VV    SSSSSS    EEEEEEEE        SSSSSS
NN  NN  NN  EEEEEEEE        TT      DD      DD  RRRRRRRR    VV      VV    SSSSSS    EEEEEEEE        SSSSSS
NN    NNNN  EE              TT      DD      DD  RR  RR      VV      VV          SS  EE                  SS
NN    NNNN  EE              TT      DD      DD  RR  RR      VV      VV          SS  EE                  SS
NN      NN  EE              TT      DD      DD  RR    RR      VV  VV            SS  EE                  SS    ....
NN      NN  EE              TT      DD      DD  RR    RR      VV  VV            SS  EE                  SS    ....
NN      NN  EEEEEEEEEE      TT      DDDDDDDD    RR      RR      VV      SSSSSSSS    EEEEEEEEEE  SSSSSSSS      ....
NN      NN  EEEEEEEEEE      TT      DDDDDDDD    RR      RR      VV      SSSSSSSS    EEEEEEEEEE  SSSSSSSS      ....


LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
0000            .TITLE  NETDRVSES - DECnet Session Control Module for NETDRIVER
0000            .IDENT  'V04-000'
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000
0000    ;++
0000    ; FACILITY:      DECNET
0000    ;
0000    ; ABSTRACT:      This module is part of NETDRIVER and is the interface between
0000    ;                the user and the NSP layer.
0000    ;
0000    ; ENVIRONMENT:  KERNEL mode, normal driver environment.
0000    ;
0000    ;--
0000
```

```
0000     .SBTTL  HISTORY
0000
0000     ;
0000     ; AUTHOR:       Alan D. Eldridge,  CREATION DATE: 11-JUN-79
0000     ;
0000     ; MODIFIED BY:
0000     ;
0000     ;       V03-022 LMP0308         L. Mark Pilant,         31-Aug-1984  16:15
0000     ;               Change default state of the ACL queue in the ORB.
0000     ;
0000     ;       V03-021 ADE1042         A. Eldridge             23-Aug-1984
0000     ;               Don't create an XWB if the RCB$W_MCOUNT is zero.  This condition
0000     ;               indicates that NETACP is going away and the test is needed to
0000     ;               avoid a race condition that can crash the system.
0000     ;
0000     ;       V03-020 ADE1041         A. Eldridge             25-Jun-1984
0000     ;               Fix loop problem in cleaning up receives.  Return SS$_CONNECFAIL
0000     ;               when an IO$_ACCESS fct can't locate the XWB (was SS$_NOLINKS).
0000     ;               Send NET$C_DR_ABORT upon IO$_DEACCESS!IO$M_ABORT (was sending
0000     ;               NET$C_DR_NORMAL).
0000     ;
0000     ;       V03-019 LMP0221         L. Mark Pilant,         7-Apr-1984  14:29
0000     ;               Change UCB$L_OWNUIC to ORB$L_OWNER and UCB$W_VPROT to
0000     ;               ORB$W_PROT.
0000     ;
0000     ;       V03-018 ADE1041         A. Eldridge     7-Mar-1984
0000     ;               Fix resource error count -- registers were screwed up.
0000     ;
0000     ;       V03-017 ADE1040         A.Eldridge      10-Sep-1983
0000     ;               Major rewrite to accomodate changes to allow NSP (NETDRVNSP.MAR)
0000     ;               to use kernel mode AST's to nibble away at the user buffers
0000     ;               rather than accessing them just at FDT or I/O post time.  This
0000     ;               change was needed to allow huge user buffers (for performance)
0000     ;               without requiring a lot of pool.
0000     ;
0000     ;
```

```
0000
0000    .SBTTL  DECLARATIONS
0000    ;
0000    ; INCLUDE FILES:
0000    ;
0000
0000            $AQBDEF
0000            $ACBDEF
0000            $CCBDEF
0000            $CRBDEF
0000            $CXBDEF
0000            $DDBDEF
0000            $DRDEF
0000            $DYNDEF
0000            $FKBDEF
0000            $IODEF
0000            $IPLDEF
0000            $IRPDEF
0000            $JIBDEF
0000            $MSGDEF
0000            $ORBDEF
0000            $PCBDEF
0000            $PHDDEF
0000            $PRDEF
0000            $SSDEF
0000            $TQEDEF
0000            $UCBDEF
0000            $VECDEF
0000
0000            $ICBDEF
0000            $IDBDEF
0000            $LPDDEF
0000            $LTBDEF
0000            $LLIDEF
0000            $RCBDEF
0000
0000            $NETSYMDEF
0000            $NETUPDDEF
0000            $NSPMSGDEF
0000
0000            $CXBEXTDEF              ; NETDRIVER CXB extensions
0000            $XWBDEF                 ; XWB and LSB definitions
0000
```

```
0000
0000    ;
0000    ; MACROS:
0000    ;
0000
0000    ;
0000    ;  Bit definition macro
0000    ;
0000    .MACRO  BITDEF  BLK,SYM,BITVAL
0000
0000            'BLK'$V_'SYM' = BITVAL
0000            'BLK'$M_'SYM' = 1@<BITVAL>
0000    .ENDM
0000
0000    ;
0000    ;  Macro to set up mailbox message filtering table
0000    ;
0000    .MACRO  MBX_FILTER      MESSAGE,BIT
0000
0000            .LONG   MBX$M_'BIT
0000            .WORD   MSG$_'MESSAGE
0000
0000    .ENDM   MBX_FILTER
0000
0000    ;
0000    ;  Macro to build a mask of XWB$M_FLG_xxx bits
0000    ;
0000    .MACRO  BLDMSK  A
0000            _$MSK   = _$MSK + XWB$M_FLG_'A'
0000    .ENDM
0000
0000    ;
0000    ;  Macro to fill the 'set' and 'clear' XWB$W_FLG tables
0000    ;
0000    .MACRO  STATEMASK       STA,SETM,CLRM
0000
0000            ;
0000            ;  Build and enter the 'set FLG' bit mask
0000            ;
0000            _$MSK = 0
0000            .IRP    A,<SETM>
0000                    BLDMSK  A
0000            .ENDR
0000            . = NET$AW_FLG_SETM + <2*XWB$C_STA_'STA'>
0000            .WORD   _$MSK
0000            ;
0000            ;  Build and enter the 'clear FLG' bit mask
0000            ;
0000            _$MSK = 0
0000            .IRP    A,<CLRM>
0000                    BLDMSK  A
0000            .ENDR
0000            . = NET$AW_FLG_CLRM + <2*XWB$C_STA_'STA'>
0000            .WORD   _$MSK
0000    .ENDM
0000
```

```
0000
0000      ;
0000      ;  Macro to initialize NSP state tables
0000      ;
0000      .MACRO  STTAB                                   ; Init state transition data
0000
0000              _$EVENT_INDEX   =  0                    ; Init event index
0000              _$ACT_INDEX     =  0                    ; Init action routine index
0000              ACT$_BUG        == 0                    ; Init the "bug-check" action routine
0000                                                      ; index
0000
0000              _$ACT_DFLT = <XWB$C_STA_CLO -           ; Default state table entry
0000                           @NET$C_ACTBITS> -          ;
0000                           + ACT$_BUG                 ;
0000
0000
0000      NET$AW_FLG_SETM:                                ; Bits to be set upon entering state
0000                      .BLKW XWB$C_NUMSTA              ; and upon timeout
0000      NET$AW_FLG_CLRM:.BLKW XWB$C_NUMSTA              ; Bits to be cleared upon entering state
0000
0000      NET$AB_STTAB:   .BLKB   0                       ; Bind the table address
0000
0000      .ENDM
0000
0000      ;
0000      ;  Macroes to move the current position within the state table
0000      ;
0000      .MACRO  ENDSTTAB                                          ; Move PC to end of table
0000              . = NET$AB_STTAB -
0000                  + <_$EVENT_INDEX * XWB$C_NUMSTA>
0000      .ENDM
0000
0000
0000      .MACRO  EVENT  EV                                         ; Setup for this event
0000
0000              EV ==  _$EVENT_INDEX                              ; Define event code
0000              .=NET$AB_STTAB + <EV * XWB$C_NUMSTA>              ; Move PC to proper event
0000              .BYTE    <_$ACT_DFLT>[XWB$C_NUMSTA]               ; Init the entry
0000                                                                ;
0000              .=NET$AB_STTAB + <EV * XWB$C_NUMSTA>              ; Move PC to proper event
0000              _$EVENT_INDEX  =  _$EVENT_INDEX + 1               ; Get ready for next event
0000
0000      .ENDM
0000
0000
0000      ;
0000      ;  Macro to fill the build and enter the state transition table element
0000      ;
0000      .MACRO  STATE   CURSTA,NXTSTA,ACTION,?LL                  ; Make table entry
0000          LL:
0000              .=.+XWB$C_STA_'CURSTA'                            ; Goto state table entry
0000              ;
0000              ;
0000              ;   If the action routine index is not defined then define the index
0000
0000              ;   Create the state-table entry.
0000              ;
```

```
0000                ;
0000                .IF NDF,ACT$_'ACTION'
0000                        ACT$_'ACTION'   = _$ACT_INDEX
0000                        _$ACT_INDEX     = _$ACT_INDEX + 1
0000                .ENDC
0000                .BYTE   <XWB$C_STA_'NXTSTA' @NET$C_ACTBITS> + ACT$_'ACTION'
0000
0000          .=LL
0000    .ENDM
0000
```

```
                    0000     244
                    0000     245 ;
                    0000     246 ; EQUATED SYMBOLS:
                    0000     247 ;
                    0000     248
                    0000     249 ;
                    0000     250 ;  Argument list offsets for QIO
                    0000     251 ;
          00000000  0000     252 P1      = 0                              ; Buffer address
          00000004  0000     253 P2      = 4                              ; Buffer length
          00000008  0000     254 P3      = 8                              ; Miscellaneous
                    0000     255
                    0000     256
                    0000     257 ASSUME  FKB$C_LENGTH  LE  ACB$C_LENGTH
                    0000     258
          00000120  0000     259 _$tmp            == <XWB$S_XWB+7>&^C<7>  ; XWB length, quad word aligned
          00000160  0000     260 XWB$$            == _$tmp+64             ; Allow enough room for the route-header
          00000120  0000     261 XWB$L_PTR_RTHD   == _$tmp                ; Ptr to route-header
          00000124  0000     262 XWB$B_ADJ_INX    == _$tmp +4             ; Adjacency index
          00000158  0000     263 XWB$T_TR3HDR     == XWB$$-8              ; Start of standard Phase III header
                    0000     264                                          ;  (must be quadword aligned)
                    0000     265
          0000017C  0000     266 XWB_C_LEN = XWB$$+ACB$C_LENGTH           ; Total XWB length
                    0000     267
                    0000     268 ;
                    0000     269 ;  Definitions for mailbox message filtering
                    0000     270 ;
                    0000     271 $VIELD MBX,0,<-
                    0000     272              <NETSTATE,,M>,-             ; Network state change
                    0000     273              <EVTAVL,,M>,-               ; Events available for logging
                    0000     274              <EVTRCVCHG,,M>,-            ; Event receiver database change
                    0000     275              <EVTXMTCHG,,M>,-            ; Event xmitter database change
                    0000     276 >
                    0000     277
                    0000     278 ;
                    0000     279 ;
                    0000     280 ;  Define a mask containing all bits indicating work needs to be done
                    0000     281 ;
                    0000     282 ;
                    0000     283 XWB$M_FLG_WMSK =        XWB$M_FLG_SCD ! -
                    0000     284                         XWB$M_FLG_SDT ! XWB$M_FLG_SDACK!-
                    0000     285                         XWB$M_FLG_SLI ! XWB$M_FLG_SIACK!-
          0000039D  0000     286                         XWB$M_FLG_CLO ! XWB$M_FLG_BREAK
                    0000     287
```

```
    0000
    0000   ;
    0000   ; DRIVER PROLOGUE TABLE
    0000   ;
    0000
00000000           .PSECT  $$$105_PROLOGUE
    0000       DPTAB   -                                   ; Define driver prologue table
    0000               END      = NET$END,-                ; End of driver
    0000               ADAPTER = NULL,-                    ; Adapter type
    0000               UCBSIZE = UCB$C LENGTH,-            ; UCB size
    0000               NAME     = NETDRIVER                ; Driver name
    0038
    0038       DPT_STORE INIT                              ; CONTROL BLOCK INIT VALUES
    0038                                                   ;
    0038       DPT_STORE CRB,CRB$L_INTD+VEC$L_ADP,L,0      ; No ADP pointer
    003F       DPT_STORE UCB,UCB$W_MB SEED,W,0             ; Init. unit value for cloning
    0044       DPT_STORE UCB,UCB$B_FIPL,B,NET$C_IPL        ; Fork IPL
    0048       DPT_STORE UCB,UCB$B_DIPL,B,NET$C_IPL        ; Device IPL
    004C       DPT_STORE ORB,ORB$B_FLAGS,B,-
                                     <ORB$M_PROT_16>   ; SOGW protection word
    0050       DPT_STORE ORB,ORB$W_PROT,W,0                ; Default protection
    0055       DPT_STORE ORB,ORB$L_OWNER,L,<^X010001>      ; Owner UIC
    005C       DPT_STORE UCB,UCB$L_DEVCHAR,L,-             ; Device characteristics
    005C                             <DEV$M_NET!-      ;   Network device
    005C                              DEV$M_AVL!-      ;   Available
    005C                              DEV$M_MBX!-      ;   Mailbox type (no hardware)
    005C                              DEV$M_IDV!-      ;   Input device
    005C                              DEV$M_ODV-       ;   Output device
    005C                             >                 ;
    0063       DPT_STORE UCB,UCB$W_DEVBUFSIZ,W,256         ;
    0068       DPT_STORE UCB,UCB$L_DEVDEPEND,L,-           ;
    0068                              MBX$M_NETSTATE   ; Enable NETSHUT by default
    006F       DPT_STORE UCB,UCB$W_STS,W,-                 ;
    006F                             <UCB$M_ONLINE!-   ; Device online
    006F                              UCB$M_TEMPLATE-; NET0 is the "template" UCB
    006F                             >                 ; used to build other NET UCBs
    0074                                                   ;
    0074       DPT_STORE REINIT                            ; CONTROL BLOCK RE-INIT VALUES
    0074
    0074       DPT_STORE DDB,DDB$L_DDT,D,NET$DDT
    0079       DPT_STORE CRB,CRB$L_INTD+VEC$L_INITIAL, D,NET$CTLR_INIT
    007E       DPT_STORE CRB,CRB$L_INTD+VEC$L_UNITINIT,D,NET$UNIT_INIT
    0083       DPT_STORE CRB,CRB$L_INTD+VEC$L_START,   D,NET$ACP COMM
    0088       DPT_STORE CRB,CRB$L_INTD+4,             D,NET$INTERRUPT
    008D
    008D       DPT_STORE END                               ;
    0000
    0000
```

```
            0000
            0000  ;
            0000  ; DRIVER DISPATCH TABLE
            0000  ;
        00000000          .PSECT  $$$115_DRIVER,LONG
            0000
            0000          DDTAB   DEVNAM  = NET,-           ; DRIVER DISPATCH TABLE
            0000                  FUNCTB  = FUNCTABLE,-     ; Function decision table address
            0000                  START   = NET$STARTIO,-   ; Start I/O operation
            0000                  ALTSTART= NET$ALTENTRY,-; Alternate I/O request entry point
            0000                  CANCEL  = NET$CANCEL,-    ; Cancel I/O entry point
            0000                  UNSOLIC = NET$UNSOL_INTR; Unsolicited interrupt
            0038
            0038
            0038  .SBTTL  FUNCTION DECISION TABLE
            0038
            0038  FUNCTABLE:                                ; FUNCTION DECISION TABLE
            0038          FUNCTAB ,-                        ; Legal Functions
            0038                  <READVBLK,READLBLK,-      ; Read
            0038                   WRITEVBLK,WRITELBLK,-    ; Write
            0038                   SETMODE,-                ; Set mailbox message filters
            0038                   ACCESS,-                 ; Logical-link Connect/Reject
            0038                   ACPCONTROL,-             ; ACP Control function
            0038                   DEACCESS,-               ; Logical-link Disconnect
            0038          >
            0040          FUNCTAB ,-                        ; BUFFERED I/O FUNCTIONS
            0040                  <READVBLK,READLBLK,-      ; Read
            0040                   WRITEVBLK,WRITELBLK,-    ; Write
            0040                   SETMODE,-                ; Set mailbox message filters
            0040                   ACCESS,-                 ; Logical-link Connect/Reject
            0040                   ACPCONTROL,-             ; ACP Control function
            0040                   DEACCESS,-               ; Logical-link Disconnect
            0040          >
            0048          FUNCTAB NET$FDT_RCV,    <READLBLK>   ; Read
            0054          FUNCTAB NET$FDT_XMT,    <WRITELBLK>  ; Write
            0060          FUNCTAB NET$FDT_ACCESS, <ACCESS>     ; Connect Logical-link
            006C          FUNCTAB NET$FDT_DEACCESS,<DEACCESS>  ; Disconnect Logical-link
            0078          FUNCTAB NET$FDT_SETMODE, <SETMODE>   ; Set mailbox message filters
            0084          FUNCTAB NET$FDT_CONTROL, <ACPCONTROL> ; ACP Control
```

```
                 0090  .SBTTL  State Table
                 0090
                 0090  ;
                 0090  ;  OWN STORAGE:
                 0090  ;
       00000080  0090          PATCH_AREA_SIZE = 128           ; Size of patch area space
                 0090
                 0090  NET$GQ_PATCH::
       00000080  0090          .LONG   PATCH_AREA_SIZE
      00000098'  0094          .LONG   .+4                     ; (not an address - offset from start
                 0098                                          ; of image to base of patch space)
       00000118  0098          .BLKB   PATCH_AREA_SIZE
                 0118
                 0118
      FFFFFEF5'  0118  NET$GL_OFF_DPTFLG::  .LONG  DPT$TAB + DPT$B_FLAGS - . ; Offset to DPT$B_FLAGS
                 011C
       00000005  011C          NET$C_ACTBITS   = 5             ; Number of action bits per entry
       00000003  011C          NET$C_STABITS   = 3             ; Number of state bits per entry
       000000E0  011C          NET$M_STAMSK    = <7>@5         ; State bit mask
                 011C
                 011C  ;
                 011C  ;
                 011C  ;  The following definitions must be contiguous to the NSPTABLES definition
                 011C  ;
                 011C  ;
                 011C  STTAB                                   ; Init state transition table
                 013C
                 013C  EVENT       NETEVT$_CI                  ; CI message received
                 013C      STATE   CIS, CIS,   BUG             ; something wrong in the driver
                 013C      STATE   CAR, CAR,   LOG             ; unexpected event
                 013C      STATE   CIR, CIR,   RCV_CR          ; Assume received retransmitted CI
                 013C      STATE   CCS, CCS,   RCV_CR          ; Assume received retransmitted CI
                 013C      STATE   RUN, RUN,   LOG             ; unexpected event
                 013C      STATE   DIS, DIS,   LOG             ; unexpected event
                 013C      STATE   DIR, DIR,   LOG             ; unexpected event
                 013C      STATE   CLO, CIR,   RCV_CI          ; inbound connect sequence
                 013C
                 013C  EVENT       NETEVT$_CA                  ; Connect Ack received
                 0144      STATE   CIS, CAR,   RCV_CA          ; measure intial round-trip time
                 0144      STATE   CAR, CAR,   NOP             ; assume retransmission
                 0144      STATE   CIR, CIR,   LOG             ; unexpected event
                 0144      STATE   CCS, CCS,   LOG             ; unexpected event
                 0144      STATE   RUN, RUN,   NOP             ; assume late arrival
                 0144      STATE   DIS, DIS,   NOP             ; assume late arrival
                 0144      STATE   DIR, DIR,   NOP             ; assume late arrival
                 0144      STATE   CLO, CLO,   NOP             ; assume late arrival
                 0144
                 0144  EVENT       NETEVT$_CC                  ; Connect Confirm received
                 014C      STATE   CIS, RUN,   RCV_CC          ; normal handshaking sequence
                 014C      STATE   CAR, RUN,   RCV_CC          ; normal handshaking sequence
                 014C      STATE   CIR, CIR,   LOG             ; unexpected event
                 014C      STATE   CCS, CCS,   LOG             ; unexpected event
                 014C      STATE   RUN, RUN,   NOP             ; assume retransmission
                 014C      STATE   DIS, DIS,   NOP             ; we enter DIS for many reasons
                 014C      STATE   DIR, DIR,   NOP             ; assume late arrival
                 014C      STATE   CLO, CLO,   RTS_NLT         ; assume late arrival
                 014C
```

```
014C
014C  EVENT     NETEVT$_PH2CCS                  ; Phase II connect confirm xmt-complete
0154      STATE     CIS, CIS,     NOP           ;
0154      STATE     CAR, CAR,     NOP           ;
0154      STATE     CIR, CIR,     NOP           ;
0154      STATE     CCS, RUN,     ENT_RUN       ; Normal Phase II handshaking sequence
0154      STATE     RUN, RUN,     NOP           ;
0154      STATE     DIS, DIS,     NOP           ;
0154      STATE     DIR, DIR,     NOP           ;
0154      STATE     CLO, CLO,     NOP           ;
0154
0154  EVENT     NETEVT$_RTS                     ; Rcv "return to sender" CI message
015C      STATE     CIS, CLO,     RCV_RTS       ; Process returned message
015C      STATE     CAR, CAR,     NOP           ; Assume late arrival on retransmission
015C      STATE     CIR, CIR,     NOP           ; Assume late arrival on retransmission
015C      STATE     CCS, CCS,     NOP           ; Assume late arrival on retransmission
015C      STATE     RUN, RUN,     NOP           ; Assume late arrival on retransmission
015C      STATE     DIS, DIS,     NOP           ; Assume late arrival on retransmission
015C      STATE     DIR, DIR,     NOP           ; Assume late arrival on retransmission
015C      STATE     CLO, CIR,     NOP           ; Assume late arrival on retransmission
015C
015C  EVENT     NETEVT$_DATA                    ; Data message received
0164      STATE     CIS, CIS,     LOG           ; unexpected event
0164      STATE     CAR, CAR,     LOG           ; unexpected event
0164      STATE     CIR, CIR,     LOG           ; unexpected event
0164      STATE     CCS, RUN,     ENT_RUN       ; a normal handshaking sequence
0164      STATE     RUN, RUN,     RCV_DATA      ; this is what NSP is for
0164      STATE     DIS, DIS,     NOP           ; unavoidable race in sending DI
0164      STATE     DIR, DIR,     NOP           ; assume late arrival
0164      STATE     CLO, CLO,     RTS_NLT       ; assume late arrival
0164
0164  EVENT     NETEVT$_DTACK                   ; Data Ack received
016C      STATE     CIS, CIS,     LOG           ; unexpected event
016C      STATE     CAR, CAR,     LOG           ; unexpected event
016C      STATE     CIR, CIR,     LOG           ; unexpected event
016C      STATE     CCS, RUN,     ENT_RUN       ; a normal handshaking sequence
016C      STATE     RUN, RUN,     RCV_DTACK     ; drive the link
016C      STATE     DIS, DIS,     NOP           ; assume late arrival or race
016C      STATE     DIR, DIR,     NOP           ; assume late arrival or race
016C      STATE     CLO, CLO,     RTS_NLT       ; assume late arrival or race
016C
```

```
016C
016C  EVENT       NETEVT$_LS                     ; Link Service msg received
0174       STATE  CIS, CIS,     LOG              ; unexpected event
0174       STATE  CAR, CAR,     LOG              ; unexpected event
0174       STATE  CIR, CIR,     LOG              ; unexpected event
0174       STATE  CCS, RUN,     ENT_RUN          ; a normal handshaking sequence
0174       STATE  RUN, RUN,     RCV_LI           ; drive the link
0174       STATE  DIS, DIS,     NOP              ; assume late arrival or race
0174       STATE  DIR, DIR,     NOP              ; assume late arrival or race
0174       STATE  CLO, CLO,     RTS_NLT          ; assume late arrival or race
0174
0174  EVENT       NETEVT$_INT                    ; Interrupt msg received
017C       STATE  CIS, CIS,     LOG              ; unexpected event
017C       STATE  CAR, CAR,     LOG              ; unexpected event
017C       STATE  CIR, CIR,     LOG              ; unexpected event
017C       STATE  CCS, RUN,     ENT_RUN          ; a normal handshaking sequence
017C       STATE  RUN, RUN,     RCV_LI           ; drive the link
017C       STATE  DIS, DIS,     NOP              ; assume late arrival or race
017C       STATE  DIR, DIR,     NOP              ; assume late arrival or race
017C       STATE  CLO, CLO,     RTS_NLT          ; assume late arrival or race
017C
017C  EVENT       NETEVT$_LIACK                  ; INT/LS Ack received
0184       STATE  CIS, CIS,     LOG              ; unexpected event
0184       STATE  CAR, CAR,     LOG              ; unexpected event
0184       STATE  CIR, CIR,     LOG              ; unexpected event
0184       STATE  CCS, RUN,     ENT_RUN          ; a normal handshaking sequence
0184       STATE  RUN, RUN,     RCV_LIACK        ; drive the link
0184       STATE  DIS, DIS,     NOP              ; assume late arrival or race
0184       STATE  DIR, DIR,     NOP              ; assume late arrival or race
0184       STATE  CLO, CLO,     RTS_NLT          ; assume late arrival or race
0184
0184  EVENT       NETEVT$_DI                     ; Disconnect Initiate msg rcv'd
018C       STATE  CIS, DIR,     RCV_Dx           ; link rejected
018C       STATE  CAR, DIR,     RCV_Dx           ; link rejected
018C       STATE  CIR, DIR,     ABORT            ; abort the link, no local owner
018C       STATE  CCS, DIR,     RCV_Dx           ; abort the link
018C       STATE  RUN, DIR,     RCV_Dx           ; abort the link
018C       STATE  DIS, DIR,     ABORT            ; change state and send DC
018C       STATE  DIR, DIR,     NOP              ; send DC
018C       STATE  CLO, CLO,     RTS_NLT          ; assume race or late arrival
018C
018C  EVENT       NETEVT$_DC                     ; Disconnect Confirm msg rcv'd
0194       STATE  CIS, CLO,     RCV_Dx           ; link rejected
0194       STATE  CAR, CLO,     RCV_Dx           ; link rejected
0194       STATE  CIR, CLO,     ABORT            ; link aborted, no local owner
0194       STATE  CCS, CLO,     RCV_Dx           ; link aborted
0194       STATE  RUN, CLO,     RCV_Dx           ; link aborted
0194       STATE  DIS, CLO,     NOP              ; normal handshaking sequence
0194       STATE  DIR, CLO,     NOP              ; assume DC is a 'no link terminate'
0194       STATE  CLO, CLO,     NOP              ; assume late arrival
0194
0194
```

```
0194
0194  EVENT       NETEVT$_DSCLNK                    ; Link failed confidence test
019C        STATE   CIS, CLO,     ABORT             ; connect timed out
019C        STATE   CAR, DIS,     ABORT             ; connect timed out
019C        STATE   CIR, DIS,     ABORT             ; local system is slow
019C        STATE   CCS, DIS,     ABORT             ; connect timed out
019C        STATE   RUN, DIS,     ABORT             ; problem talking with remote node
019C        STATE   DIS, CLO,     NOP               ; abort the link
019C        STATE   DIR, CLO,     NOP               ; abort the link
019C        STATE   CLO, CLO,     NOP               ; Try to deallocate XWB
019C
019C
019C  EVENT       NETEVT$_CANLNK                    ; Local cancel of link
01A4        STATE   CIS, CLO,     CANLNK            ; abort from a Connect state
01A4        STATE   CAR, CLO,     CANLNK            ; abort from a Connect state
01A4        STATE   CIR, DIS,     CANLNK            ; abort link, no local owner
01A4        STATE   CCS, DIS,     CANLNK            ; abort from a Connect state
01A4        STATE   RUN, DIS,     CANLNK            ; orderly shuttdown
01A4        STATE   DIS, DIS,     NOP               ; link is already disconnecting
01A4        STATE   DIR, DIR,     NOP               ; link is already disconnecting
01A4        STATE   CLO, CLO,     NOP               ; link is already disconnecting
01A4
01A4  EVENT       NETEVT$_RESDIS                    ; Resume disconnect
01AC        STATE   CIS, CIS,     BUG               ; Valid only from RUN state
01AC        STATE   CAR, CAR,     BUG               ; Valid only from RUN state
01AC        STATE   CIR, CIR,     BUG               ; Valid only from RUN state
01AC        STATE   CCS, CCS,     BUG               ; Valid only from RUN state
01AC        STATE   RUN, DIS,     RES_DISC          ; Disconnect if XWB is idle
01AC        STATE   DIS, DIS,     BUG               ; Valid only from RUN state
01AC        STATE   DIR, DIR,     BUG               ; Valid only from RUN state
01AC        STATE   CLO, CLO,     BUG               ; Valid only from RUN state
01AC
01AC
```

```
01AC
01AC  EVENT     NETEVT$_CIA                   ; Connect Initiate IO$_ACCESS
01B4        STATE   CIS, CIS,     BUG          ; XWB was just created
01B4        STATE   CAR, CAR,     BUG          ; XWB was just created
01B4        STATE   CIR, CAR,     BUG          ; XWB was just created
01B4        STATE   CCS, CCS,     BUG          ; XWB was just created
01B4        STATE   RUN, RUN,     BUG          ; XWB was just created
01B4        STATE   DIS, DIS,     BUG          ; XWB was just created
01B4        STATE   DIR, DIR,     BUG          ; XWB was just created
01B4        STATE   CLO, CIS      INITIATE     ; Normal connect initiate seq.
01B4
01B4  EVENT     NETEVT$_CCA                   ; Connect Confirm IO$_ACCESS
01BC        STATE   CIS, CIS,     SSABORT      ; Confirm not possible
01BC        STATE   CAR, CAR,     SSABORT      ; Confirm not possible
01BC        STATE   CIR, CCS,     CONFIRM      ; Normal connect confirm seq.
01BC        STATE   CCS, CCS,     SSABORT      ; Confirm not possible
01BC        STATE   RUN, RUN,     SHRLNK       ; Second accessor to link
01BC        STATE   DIS, DIS,     SSABORT      ; Confirm no longer possible
01BC        STATE   DIR, DIR,     SSABORT      ; Confirm no longer possible
01BC        STATE   CLO, CLO,     SSABORT      ; Confirm no longer possible
01BC
01BC  EVENT     NETEVT$_CRA                   ; Connect Reject IO$_ACCESS
01C4        STATE   CIS, CIS,     SSABORT      ; Reject not possible
01C4        STATE   CAR, CAR,     SSABORT      ; Reject not possible
01C4        STATE   CIR, DIS,     CONFIRM      ; Normal connect reject seq.
01C4        STATE   CCS, CCS,     SSABORT      ; Reject not possible
01C4        STATE   RUN, RUN,     SSABORT      ; Reject not possible
01C4        STATE   DIS, DIS,     SSABORT      ; Reject not possible
01C4        STATE   DIR, DIR,     SSABORT      ; Reject not possible
01C4        STATE   CLO, CLO,     SSABORT      ; Reject not possible
01C4
01C4  EVENT     NETEVT$_DEA                   ; QIO IO$_DEACCESS
01CC        STATE   CIS, CIS,     BUG          ; Channel should not have window
01CC        STATE   CAR, CAR,     BUG          ; Channel should not have window
01CC        STATE   CIR, CIR,     BUG          ; Channel should not have window
01CC        STATE   CCS, CCS,     BUG          ; Channel should not have window
01CC        STATE   RUN, DIS,     DEACCESS     ; But change to DIS state only
01CC                                           ;  if this is the last accessor
01CC        STATE   DIS, DIS,     DEACCESS     ; Link was aborted externally
01CC        STATE   DIR, DIR,     DEACCESS     ; Link was aborted externally
01CC        STATE   CLO, CLO,     DEACCESS     ; Link was aborted externally
01CC
```

```
01CC
01CC   EVENT      NETEVT$_MBXERR               ; Fatal error writing to mailbox
01D4        STATE   CIS, CLO,     ABORT        ; abort from a Connect state
01D4        STATE   CAR, CLO,     ABORT        ; abort from a Connect state
01D4        STATE   CIR, DIS,     ABORT        ; abort link, no local owner
01D4        STATE   CCS, DIS,     ABORT        ; abort from a Connect state
01D4        STATE   RUN, DIS,     ABORT        ; abort from the RUN state
01D4        STATE   DIS, DIS,     NOP          ; link is already disconnecting
01D4        STATE   DIR, DIR,     NOP          ; link is already disconnecting
01D4        STATE   CLO, CLO,     NOP          ; link is already disconnecting
01D4
01D4   EVENT      NETEVT$_PROERR               ; Protocol error     (NOP for now)
01DC        STATE   CIS, CIS,     NOP          ;
01DC        STATE   CAR, CAR,     NOP          ;
01DC        STATE   CIR, DIS,     NOP          ;
01DC        STATE   CCS, DIS,     NOP          ;
01DC        STATE   RUN, RUN,     NOP          ;
01DC        STATE   DIS, DIS,     NOP          ;
01DC        STATE   DIR, DIR,     NOP          ;
01DC        STATE   CLO, CLO,     NOP          ;
01DC
01DC
```

```
          01DC
          01DC   ;
          01DC   ;  Setup tables which specify which XWB$W_FLG bits to set and clear upon
          01DC   ;  a transition into a new state.
          01DC   ;
          01DC   ;         New
          01DC   ;         State  Flags to set      Flags to clear
          01DC   ;         -----  ------------      --------------
          01DC   ;
          01DC   STATEMASK CIS,  <SCD>              <WBUF>
          0130   STATEMASK CAR,  <CLO>              <WBUF>
          0132   STATEMASK CIR,  <SCD>              <WBUF>
          0134   STATEMASK CCS,  <SCD>              <WBUF>
          0136   STATEMASK RUN,  <SDT,SDFL,WHGL>    <WBUF,SCD>
          0138   STATEMASK DIR,  <SCD>              <WBUF,WBP,WHGL,WDAT,SDT,SLI,SDACK,SIACK,-
          0138                                                      BREAK,IAVL,TBPR,SIFL,SDFL>
          013A   STATEMASK DIS,  <SCD>              <WBUF,WBP,WHGL,WDAT,SDT,SLI,SDACK,SIACK,-
          013A                                                      BREAK,IAVL,TBPR,SIFL,SDFL>
          013C   STATEMASK CLO,  <CLO>              <WBUF,WBP,WHGL,WDAT,SDT,SLI,SDACK,SIACK,-
          013C                                                  SCD,BREAK,IAVL,TBPR,SIFL,SDFL>
          012E
          012E   ENDSTTAB
          01E4
          01E4
          01E4   ;
          01E4   ; The following mask is used to identify the subset of flags used
          01E4   ; to signal work to be done
          01E4   ;
0000039D  01E4   NET$GL_WORKBITS:: .LONG XWB$M_FLG_WMSK  ; Flags requiring work to be done
          01E8
          01E8
          01E8   MBX_TABLE:                                  ; Table for mapping mbx msg codes
          01E8                                               ; to filter bits
          01E8           MBX_FILTER      NETSHUT,NETSTATE    ; Network state change
          01EE           MBX_FILTER      EVTAVL,EVTAVL       ; Events available for logging
          01F4           MBX_FILTER      EVTRCVCHG,EVTRCVCHG ; Event receiver database change
          01FA           MBX_FILTER      EVTXMTCHG,EVTXMTCHG ; Event xmitter database change
00000000  0200           .LONG   0                           ; End of table
          0204
```

```
                    0204     .SBTTL  NET$AZ_DR_TABLE - Disconnect Reason Code Mapping
                    0204
                    0204
                    0204     ;
                    0204     ;  Macro to set up connect reject reason codes
                    0204     ;
          00000000  0204     REASON_W_DR     == 0                 ; Reason code
          00000002  0204     REASON_W_SS     == 2                 ; SS$_.. to return in data IRP's
          00000004  0204     REASON_W_MBX    == 4                 ; MBX$_.. message code
          00000006  0204     REASON_C_LENGTH == 6
                    0204
                    0204     .MACRO  MRC     REASON,SS_CODE,MSG_CODE
                    0204
                    0204             .WORD   NET$C_DR_'REASON
                    0204             .WORD   SS$_'SS_CODE
                    0204             .WORD   MSG$_'MSG_CODE
                    0204
                    0204     .ENDM   MRC
                    0204
                    0204
          00000064  0204     NET$C_DR_INVALID   == 100                     ; Fake value meaning "not setup"
          00000066  0204     NET$C_DR_DEACC     == 102                     ; Fake value for code conversion
                    0204
                    0204     NET$AZ_DR_TABLE:                              ; Table for mapping disconnect reasons
                    0204                                                   ; for other than the "connect-initiate"
                    0204                                                   ; state
                    0204     ;
                    0204     ;
                    0204     ;     discon  data          mailbox
                    0204     ;     reason  status        message
                    0204     ;     ------  ----------    --------
                    0204     ;
                    0204     MRC   NORMAL, LINKDISCON, DISCON               ;
                    020A     MRC   EXIT,   LINKEXIT,   EXIT                 ; User exit or timeout
                    0210     MRC   NOPATH, PATHLOST,   PATHLOST             ; Path lost to partner node
                    0216     MRC   SHUT,   SHUT,       NETSHUT              ; Node shutting down
                    021C     MRC   NOBJ,   PROTOCOL,   ABORT                ; No such object
                    0222     MRC   ABORT,  LINKABORT,  ABORT                ; Disconnect abort
                    0228     MRC   THIRD,  THIRDPARTY, THIRDPARTY           ; Disconnect by third party
                    022E     MRC   ACCESS, PROTOCOL,   ABORT                ; Login info invalid
                    0234     MRC   RSU,    PROTOCOL,   ABORT                ; Resource error
                    023A     MRC   BUSY,   PROTOCOL,   ABORT                ; Object too busy
                    0240     MRC   FMT,    PROTOCOL,   ABORT                ; Illegal process name field
                    0246     MRC   NONODE, PROTOCOL,   ABORT                ; Unrecognized node i.d.
                    024C     MRC   IVNODE, PROTOCOL,   ABORT                ; Invalid node-i.d. format
                    0252           ;
                    0252           ;
                    0252           ;  The following are internal codes and are not part of NSP
                    0252           ;
                    0252           ;
                    0252     MRC   DEACC,  LINKDISCON, DISCON               ; Link is IO$_DEACCESS'ed
                    0258     MRC   INVALID,LINKABORT,  ABORT                ; Reason field never setup
                    025E
          FFFFFFFF  025E     .LONG -1                                      ; Terminate the table (the last entry
                    0262                                                   ; is to be used as a catch-all)
                    0262
                    0262
```

```
                        0262     NET$AZ_DR_CONTAB:                                     ; Table for mapping reject reasons
                        0262                                                           ; in one of the "connect" states
                        0262     ;
                        0262     ;
                        0262     ;       discon  connect         mailbox
                        0262     ;       reason  status          message
                        0262     ;       ------  -----------     --------
                        0262     ;
                        0262     MRC     NORMAL, REJECT,         REJECT                ; Connect reject
                        0268     MRC     EXIT,   LINKEXIT,       EXIT                  ; User exit or timeout
                        026E     MRC     NOPATH, UNREACHABLE,    PATHLOST              ; Path lost to partner node
                        0274     MRC     SHUT,   SHUT,           NETSHUT               ; Node shutting down
                        027A     MRC     NOBJ,   NOSUCHOBJ,      REJECT                ; No such object
                        0280     MRC     ABORT,  LINKABORT,      ABORT                 ; Disconnect abort
                        0286     MRC     THIRD,  THIRDPARTY,     THIRDPARTY            ; Disconnect by third party
                        028C     MRC     ACCESS, INVLOGIN,       REJECT                ; Login info invalid
                        0292     MRC     RSU,    REMRSRC,        REJECT                ; Resource error
                        0298     MRC     BUSY,   REMRSRC,        REJECT                ; Object too busy
                        029E     MRC     FMT,    PROTOCOL,       REJECT                ; Illegal process name field
                        02A4     MRC     NONODE, NOSUCHNODE,     REJECT                ; Unrecognized node i.d.
                        02AA     MRC     IVNODE, NOSUCHNODE,     REJECT                ; Invalid node-i.d. format
                        02B0             ;
                        02B0             ;
                        02B0             ;  The following are internal codes and are not part of NSP
                        02B0             ;
                        02B0             ;
                        02B0     MRC     DEACC,  ABORT,          ABORT                 ; Link is IO$_DEACCESS'ed
                        02B6     MRC     INVALID,CONNECFAIL,     ABORT                 ; Reason field never setup
                        02BC
               FFFFFFFF 02BC     .LONG -1                                              ; Terminate the table (the last entry
                        02C0                                                           ; is to be used as a catch-all)
                        02C0
                        02C0
                        02C0
                        02C0     NET$MAP_R_REASON::                                    ; Map Reason code in XWB$W_R_REASON
50    FF3A CF     9E    02C0                   MOVAB    W^NET$AZ_DR_TABLE -             ; Setup non-connect table address
                        02C5                             -REASON_C_LENGTH,R0          ; ...biased for scan
                        02C5
                        02C5                   ASSUME   XWB$C_STA_CLO  EQ  0
                        02C5                   ASSUME   XWB$C_STA_CIS  EQ  1
                        02C5                   ASSUME   XWB$C_STA_CAR  EQ  2
                        02C5
  1E A5     02    91    02C5                   CMPB     #2,XWB$B_STA(R5)               ; Is 'connect intiate' table needed?
            05    19    02C9                   BLSS     10$                            ; If LSS then no
50    FF8D CF     9E    02CB                   MOVAB    W^NET$AZ_DR_CONTAB -           ; Setup connect-intiate table address
                        02D0                             -REASON_C_LENGTH,R0          ; ...biased for scan
                        02D0                                                           ;
      50    06    C0    02D0     10$:          ADDL     #REASON_C_LENGTH,R0            ; Goto next entry
         44 A5    B1    02D3                   CMPW     XWB$W_R_REASON(R5),-           ; Does it match ?
            60          02D6                            REASON_Q_DR(R0)                ;
            07    13    02D7                   BEQL     20$                            ; If EQL then yes
            60    D5    02D9                   TSTL     (R0)                           ; At end of table?
            F3    18    02DB                   BGEQ     10$                            ; If GEQU then no
      50    06    C2    02DD                   SUBL     #REAS(^N_C_LENGTH,R0           ; No match found, use the default entry
                  05    02E0     20$:          RSB
                        02E1
```

```
        02E1   785
        02E1   786
        02E1    788
        02E1    789 NET$INTERRUPT:
        02E1    790 NET$CTLR_INIT:
  05    02E1    791          RSB
        02E2    792 NET$UNIT_INIT:
  01    02E2    793          NOP
  01    02E3    794          NOP
  01    02E4    795          NOP
  01    02E5    796          NOP
  01    02E6    797          NOP
  01    02E7    798          NOP
  01    02E8    799          NOP
  01    02E9    800          NOP
  01    02EA    801          NOP
  05    02EB    802          RSB
        02EC    803
```

```
                                02EC   .SBTTL  NET$FORK        - Fork the XWB to do new work
                                02EC   ;+
                                02EC   ;
                                02EC   ;  If the fork block in the XWB is available, it is forked so that the work
                                02EC   ;  in XWB$W_FLG will be done.  If the fork block is unavailable, no further
                                02EC   ;  action is required since the XWB$W_FLG work will get done when to XWB fork
                                02EC   ;  process is subsequently resumed.
                                02EC   ;
                                02EC   ;
                                02EC   ;  INPUTS:       R5      XWB address
                                02EC   ;                R0      Garbage
                                02EC   ;
                                02EC   ;  OUTPUTS:      R0      #1
                                02EC   ;
                                02EC   ;                All other registers are preserved
                                02EC   ;
                                02EC   ;-
                                02EC   NET$FORK::                                                     ; Fork the XWB
        06 0E A5    02     E2   02EC           BBSS    #XWB$V_STS_SOL,XWB$W_STS(R5),20$               ; If BS, fork block in use
                                02F1
                    3E     BB   02F1           PUSHR   #^M<R1,R2,R3,R4,R5>                            ; Save regs
                    06     10   02F3           BSBB     30$                                           ; Schedule fork and return
                    3E     BA   02F5   10$:    POPR    #^M<R1,R2,R3,R4,R5>                            ;
                                02F7                                                                  ;
              50    01     D0   02F7   20$:    MOVL    #1,R0                                          ; Always return success
                           05   02FA           RSB                                                    ; Done
                                02FB
           55    14 A5     9E   02FB   30$:    MOVAB    XWB$Q_FORK(R5),R5                             ; Switch to fork block context
          00000000'GF      16   02FF           JSB      G^EXE$FORK                                    ; Fork
           55    EC A5     9E   0305           MOVAB   -XWB$Q_FORK(R5),R5                             ; Restore XWB context
                                0309
           0E A5    04     AA   0309           BICW    #XWB$M_STS_SOL,XWB$W_STS(R5)                   ; We're back
           1C A5    02     AA   030D           BICW    #XWB$M_FLG_WBUF,XWB$Q_FLG(R5)                  ; Clear wait flag to allow retry
        10 0E A5    03     E1   0311           BBC     #XWB$V_STS_DIS,XWB$W_STS(R5),100$             ; If BC, disconnect not pending
                                0316
           0FC0 8F         BB   0316           PUSHR    #^M<R6,R7,R8,R9,R10,R11>                      ; Save Event regs
                    5B     D4   031A           CLRL     R11                                           ; Say "not okay to go to IPL 2"
              57    0E     9A   031C           MOVZBL   #NETEVT$_RESDIS,R7                            ; Event is "resume deaccess"
                  0034     30   031F           BSBW     NET$EVENT                                     ; Signal the event
           0FC0 8F         BA   0322           POPR     #^M<R6,R7,R8,R9,R10,R11>                      ; Restore regs
                                0326
                  007A     30   0326   100$:   BSBW     NET$SCH_MSG                                   ; Schedule message transmission
                           05   0329           RSB                                                    ; Done
                                032A
                                032A
```

```
                        032A   .SBTTL  NET$END_EVENT   - Abort current event without changing state
                        032A   .SBTTL  NET$COMPLEX_EV  - Change state and process new event
                        032A   .SBTTL  NET$PRE_EMPT    - Process new event without changing state
                        032A   ;+
                        032A   ;
                        032A   ;  These routines are called by the dispatched event action routines in order
                        032A   ;  to complete current event processing in a non-standard way.  They should be
                        032A   ;  considered substitutes to the RSB instruction which is normally used to
                        032A   ;  return control -- consequently the stack is checked for the return address
                        032A   ;  of the event dispatcher.
                        032A   ;
                        032A   ;  CALLING SEQUENCE:    JMP  NET$xxx
                        032A   ;
                        032A   ;  INPUTS: R10     Preserved for call to action routine
                        032A   ;          R9      The value originally stored by the event dispatcher
                        032A   ;          R8      Preserved for call to action routine
                        032A   ;          R7      Code of new event to be processed (scratch if NET$END_EVENT)
                        032A   ;          R6      The value originally stored by the event dispatcher
                        032A   ;          R5      XWB address
                        032A   ;          R4-R1   Scratch
                        032A   ;          R0      If NET$END_EVENT the status to be returned to the
                        032A   ;                     caller of the event dispatcher,
                        032A   ;                  Else scratch
                        032A   ;        (SP)      The address of CHANGE_STA which is the NET$EVENT return
                        032A   ;                  address.
                        032A   ;
                        032A   ;  OUTPUTS:  N/A
                        032A   ;
                        032A   ;-
                        032A   NET$END_EVENT::                                 ; End event without changing state
            19    10    032A           BSBB    CHKRETADDR                      ; Make sure stack is setup properly
          0074    30    032C           BSBW    NET$SCH_MSG                     ; Schedule message transmission
                  05    032F           RSB
                        0330
                        0330   NET$COMPLEX_EV::                                ; Change state, process new event
            13    10    0330           BSBB    CHKRETADDR                      ; Validate state of stack
            05    EF    0332           EXTZV   #NET$C_ACTBITS,-                ;
    54   59    03       0334                   #NET$C_STABITS,R9,R4            ; Get next state
   1E A5   54    91     0337           CMPB    R4,XWB$B_STA(R5)                ; New state ?
            02    13    033B           BEQL    10$                             ; If not, branch
            3F    10    033D           BSBB    NEW_STATE                       ; Enter new state
            15    11    033F   10$:    BRB     NET$EVENT                       ; Process new event
                        0341                                                   ;
                        0341   NET$PRE_EMPT::                                  ; Pre-empt the current event
            02    10    0341           BSBB    CHKRETADDR                      ; Validate state of stack
            11    11    0343           BRB     NET$EVENT                       ; Process new event
                        0345
                        0345   CHKRETADDR:                                     ; Checks return address to trap bugs
        6F'AF     9F    0345           PUSHAB  B^CHANGE_STA                    ; Prepare for next instruction
   04 AE   8E     D1    0348           CMPL    (SP)+,4(SP)                     ; Is state of stack correct?
            04    12    034C           BNEQ    5$                              ; If NEQ then no
       6E   8E    D0    034E           MOVL    (SP)+,(SP)                      ; Overlay return address
                  05    0351           RSB                                     ; Return
                        0352
                        0352   5$:     BUG_CHECK NETNOSTATE,FATAL
                        0356
```

```
                                0356  .SBTTL  NET$EVENT       - Event dispatcher
                                0356  ;+
                                0356  ;
                                0356  ;  This is the state table event dispatcher used to determine what is to be
                                0356  ;  done and what state the XWB is to enter next.  An event only has meaning
                                0356  ;  within the context of a XWB.
                                0356  ;
                                0356  ;
                                0356  ;  INPUTS:         R10     Preserved for call to action routine
                                0356  ;                  R9      Available for the event dispatcher's exclusive use
                                0356  ;                  R8      Preserved for call to action routine
                                0356  ;                  R7      Code of the event to be processed
                                0356  ;                  R6      If received message event then Tranport's IRP address
                                0356  ;                          If "startio" event then UCB address
                                0356  ;                  R5      Address of XWB
                                0356  ;                  R4      Scratch
                                0356  ;                  R3      If received message event then scratch
                                0356  ;                          If "startio" event then QIO IRP address
                                0356  ;                  R2      If received msg event then message bytes not yet accounted f
                                0356  ;                          If "startio" event then scratch
                                0356  ;                  R1      If received msg event then ptr to first unprocessed byte in
                                0356  ;                          If "startio" event then scratch
                                0356  ;                  R0      Scratch
                                0356  ;
                                0356  ;  OUTPUTS:        R0      Status code from the action routine to be returned to
                                0356  ;                          the caller of the event dispatcher.
                                0356  ;
                                0356  ;                       Only R6 and R5 are preserved.
                                0356  ;
                                0356  ;-
                                0356
                                0356          ASSUME  XWB$C_NUMSTA EQ 8       ; Assume quadword per event
                                0356  NET$EVENT::                             ; Process new event
            59   1E A5      9A  0356          MOVZBL  XWB$B_STA(R5),R9        ; Get current state
       54   FDDD CF47       7E  035A          MOVAQ   NET$AB_STTAB[R7],R4     ; Get event block address
            59   6449       9A  0360          MOVZBL  (R4)[R9],R9             ; Get table entry
  54   59   000000E0 8F     CB  0364          BICL3   #NET$M_STAMSK,R9,R4     ; Get action routine index
                                036C          ;
                                036C          ;
                                036C          ;       Dispatch according to the event code.  The action routines
                                036C          ;       can assume the following :
                                036C          ;
                                036C          ;       Inputs:
                                036C          ;
                                036C          ;          R10     Parameter from caller to action routine
                                036C          ;          R9      State information -- not to be touched
                                036C          ;          R8      Parameter from caller to action routine
                                036C          ;          R7      Event code
                                036C          ;          R6      Varies with event
                                036C          ;          R5      XWB address
                                036C          ;          R4      Scratch
                                036C          ;          R3-R1   Varies with event
                                036C          ;          R0      Scratch
                                036C          ;        (SP)      Return address
                                036C          ;
                                036C          ;       Returned values:
                                036C          ;
                                036C   965          ;       R8,R7   Garbage
                                036C   966          ;       R6,R5   Preserved
                                036C   967          ;       R4-R1   Garbage
                                036C   968          ;       R0      Status to be returned to caller of dispatcher
                                036C   969          ;
                                036C   970          ;
                  00D8     30   036C   971          BSBW    ACT_DISPATCH            ; Call action routine
                                036F   972
                                036F   973  CHANGE_STA:                             ; Change logical-link state
                    05     EF   036F   974          EXTZV   #NET$C_ACTBITS,-        ;
              54 59 03          0371   975                  #NET$C_STABITS,R9,R4    ; Get next state
           1E A5    54     91   0374   976          CMPB    R4,XWB$B_STA(R5)        ; New state ?
                    29     13   0378   977          BEQL    NET$SCH_MSG             ; If EQL no, schedule message xmission
                    02     10   037A   978          BSBB    NEW_STATE               ; Change to new state
                    25     11   037C   979          BRB     NET$SCH_MSG             ; Schedule message xmission
                                037E   980
                                037E   981
                                037E   982  NEW_STATE:                              ; Change to new logical-link state
                                037E   983          ;
                                037E   984          ;
                                037E   985          ;    Clear PROGRESS since we are changing states.  The only exceptions
                                037E   986          ;    are if we are coming out of the "closed" state (since PROGRESS has
                                037E   987          ;    been setup to the correct value by the previous action routine) or
                                037E   988          ;    if we are entering the "connect ACK received" state (since we the
                                037E   989          ;    outgoing link timeout period is tied to the receipt of a Connect
                                037E   990          ;    Confirm, not a Connect-ACK).
                                037E   991          ;
                                037E   992          ;
                    00     91   037E   993          CMPB    #XWB$C_STA_CLO,-        ; Coming out of the "closed" state?
                 1E A5          0380   994                   XWB$B_STA(R5)
                    08     13   0382   995          BEQL    10$                     ; If EQL, PROGRESS was already init'd
              54    02     91   0384   996          CMPB    #XWB$C_STA_CAR,R4       ; Entering "CAR" state?
                    03     13   0387   997          BEQL    10$                     ; If EQL yes, do not re-init PROGRESS
                 52 A5     B4   0389   998          CLRW    XWB$W_PROGRESS(R5)      ; Init progress count
           1E A5    54     90   038C   999  10$:    MOVB    R4,XWB$B_STA(R5)        ; Change state
                           AA   0390  1000          BICW    #XWB$M_STS_TID!-        ; Always clear "timer id valid" and
                                0391  1001                   XWB$M_STS_DIS,-        ; "disconnect pending" flags
           0E A5    09          0391  1002                   XWB$W_STS(R5)          ;
                                0394  1003                                                  ;
   1C A5    FD93 CF44      AA   0394  1004          BICW    NET$AW_FLG_CLRM[R4],XWB$W_FLG(R5) ; Clear indicated flags
   1C A5    FD7C CF44      A8   039B  1005          BISW    NET$AW_FLG_SETM[R4],XWB$W_FLG(R5) ; Set indicated flags
                           05   03A2  1006          RSB                                       ; Done
                                03A3  1007
```

```
                                      03A3  .SBTTL  NET$SCH_MSG     - schedule message transmission
                                      03A3  ;+
                                      03A3  ;
                                      03A3  ;  The following flags are used to cause control messages to be setup when the
                                      03A3  ;  control message cell in the XWB becomes available.  As each message is
                                      03A3  ;  entered into this contol message cell, the corresponding bit is cleared.
                                      03A3  ;
                                      03A3  ;  These flags are listed in the order of their priority.
                                      03A3  ;
                                      03A3  ;
                                      03A3  ;       XWB$V_FLG_TBPR - Set whenever the receive back pressure state needs to
                                      03A3  ;                        be toggled.
                                      03A3  ;
                                      03A3  ;       XWB$V_FLG_IAVL - Set whenever a new xmit interrupt IRP makes it to the
                                      03A3  ;                        head of the LSB queue and the partner's flow control
                                      03A3  ;                        on the INT/LS subchannel will let us send the message.
                                      03A3  ;
                                      03A3  ;       XWB$V_FLG_SIFL - Set whenever an INTERRUPT message has been sent to the
                                      03A3  ;                        user's mailbox.
                                      03A3  ;
                                      03A3  ;       XWB$V_FLG_SDFL - Set whenever the inactivity timer fires in order to
                                      03A3  ;                        maintain a minimal amount of traffic on the link to
                                      03A3  ;                        see if the remote node is still active.
                                      03A3  ;
                                      03A3  ;
                                      03A3  ;  Whether or not a new Link-service/Interrupt message is setup in the XWB
                                      03A3  ;  cell, XWB$W_FLG(R5) is scanned to see if any work needs to be done.  If
                                      03A3  ;  so, and if the XWB fork block is not in use, control is passed to
                                      03A3  ;  NET$SOLICIT.
                                      03A3  ;
                                      03A3  ;
                                      03A3  ;  INPUTS:       R5      XWB address
                                      03A3  ;                R4-R0   Scratch
                                      03A3  ;
                                      03A3  ;  OUTPUTS:      R4-R0   Garbage
                                      03A3  ;
                                      03A3  ;                All other registers are preserved.
                                      03A3  ;
                                      03A3  ;-
                                      03A3  NET$SCH_MSG::                                           ; Schedule message transmission
                                      03A3
                                      03A3          ASSUME  XWB$V_FLG_IAVL  EQ  1+XWB$V_FLG_TBPR
                                      03A3          ASSUME  XWB$V_FLG_SIFL  EQ  1+XWB$V_FLG_IAVL
                                      03A3          ASSUME  XWB$V_FLG_SDFL  EQ  1+XWB$V_FLG_SIFL
                                      03A3
 50   1C A5    04    0B    EA         03A3          FFS     #XWB$V_FLG_TBPR,#4,XWB$W_FLG(R5),R0 ; Find message to build
                     74    13         03A9          BEQL     90$                                ; If EQL then none
      6F 6C A5    04    E0            03AB          BBS     #NSP$V_FLW_INUSE,XWB$B_X_FLW(R5),90$; If BS, msg cell is in use
         6C A5    10    90            03B0          MOVB    #NSP$M_FLW_INUSE,XWB$B_X_FLW(R5)    ; Claim the cell, clear flags
            6D A5    94               03B4          CLRB     XWB$B_X_FLWCNT(R5)                 ; Init flow request count
                                      03B7                                                      ;
      52   00D4 C5    9E              03B7          MOVAB    XWB$T_LI(R5),R2                    ; Setup LSB pointer
                     62    B6         03BC          INCW     LSB$W_LUX(R2)                      ; Get next sequence number
      62   F000 8F    AA              03BE          BICW    #^X<F000>,LSB$W_LUX(R2)             ; Mask off junk bits
         04 A2    62    B0            03C3          MOVW     LSB$W_LUX(R2),LSB$W_HXS(R2)        ; It's sendable now
      00 1C A5    50    E5            03C7          BBCC     R0,XWB$W_FLG(R5),20$               ; Clear the work bit
                                      03CC                                                      ;
```

```
                        03CC 20$:  $DISPATCH  R0,-                               ; Dispatch on work bit
                        03CC       <-
                        03CC         <XWB$V_FLG_IAVL,   50$>,-                   ; INTerrupt msg
                        03CC         <XWB$V_FLG_SIFL,   40$>,-                   ; INTerrupt flow control msg
                        03CC         <XWB$V_FLG_SDFL,   80$>,-                   ; DATA flow control msg
                        03CC       >                                             ; Fall thru if
                        03D6                                                     ;    R0=XWB$V_FLG_TBPR
                        03D6       ;
                        03D6       ;
                        03D6       ;    Setup for DATA channel control message to toggle the local
                        03D6       ;    receiver back pressure state
                        03D6       ;
                        03D6       ;
       50    01    90   03D6       MOVB    #NSP$M_FLW_XOFF,R0                    ; Setup for "stop flow" message
  13 0E A5   06    E3   03D9       BBCS    #XWB$V_STS_RBP,XWB$W_STS(R5),30$ ; If BC, not back-pressured off
 0E A5     0040 8F AA   03DE       BICW    #XWB$M_STS_RBP,XWB$W_STS(R5)     ; Mark receiver as having its
                        03E4                                                ; back-pressured relaxed
       50    02    90   03E4       MOVB    #NSP$M_FLW_XON,R0                     ; Setup for "start flow" msg
                        03E7       ;
                        03E7       ;
                        03E7       ;    Force a NAK on the DATA sub-channel in order to reset its sequence
                        03E7       ;    numbers.  Ordinarily, NAK's are undesireable in a routing
                        03E7       ;    environment since they could contribute to congestion problems.
                        03E7       ;    But that is generally true for NAK's sent due to receiving messages
                        03E7       ;    out of order (a message received out of order is often due to
                        03E7       ;    network congestion loss of an earlier packet).
                        03E7       ;
                        03E7       ;    The NAK is not absolutely necessary, but failure to send it will
                        03E7       ;    mean an inordinate delay since the remote sequence numbers will
                        03E7       ;    not be reset when back-pressure is subsequently relaxed (only a
                        03E7       ;    NAK or timeout resets the sequence numbers).
                        03E7       ;
                        03E7       ;       NOTE:   Since XWB$M_FLG_SDACK is less than XMB$M_FLG_SLI, the
                        03E7       ;               NAK will be sent just before the back-pressure message.
                        03E7       ;               This is the desired order.
                        03E7       ;
                        03E7       ;
                        03E7       ;
    1C A5    08    A8   03E7       BISW    #XWB$M_FLG_SDACK,XWB$W_FLG(R5)       ; Force NAK on the DATA channel
 0E A5     0100 8F A8   03EB       BISW    #XWB$M_STS_DTNAK,XWB$W_STS(R5)       ;   in order to reset it
    6C A5    50    88   03F1 30$:  BISB     R0,XWB$B_X_FLW(R5)                  ; Set remaining control flags
 1B 1C A5    0D    E5   03F5       BBCC    #XWB$V_FLG_SIFL,XWB$W_FLG(R5),80$; Piggy-back INT flow control
                        03FA                                                ; message if possible
                        03FA 40$:  ;
                        03FA       ;
                        03FA       ;    Setup Interrupt flow-control message.
                        03FA       ;
                        03FA       ;
    6C A5    04    88   03FA       BISB    #NSP$M_FLW_LISUB,XWB$B_X_FLW(R5) ; Flow control for LI channel
       6D A5       96   03FE       INCB     XWB$B_X_F[WCN1(R5)                  ; Ask for one more INT message
     00FD C5       96   0401       INCB     XWB$T_LI+LSB$B_R_CXBQUO(R5)         ; And allow it to be received
          0E       11   0405       BRB      80$                                 ; Schedule msg for transmission
                        0407 50$:  ;
                        0407       ;
                        0407       ;    Setup for Interrupt message
                        0407       ;
                        0407       ;
```

```
        6C A5   20   88  0407            BISB    #NSP$M_FLW_INT,XWB$B_X_FLW(R5)      ; Not ''link service'' message
           50   10 A2 D0  040B           MOVL     LSB$L_X_PND(R2),R0                 ; Get IRP
  20 A0    0040 8F   AA  040F            BICW    #IO$M_INTERRUPT,IRP$W_FUNC(R0)      ; Indicate state change
                         0415      80$:  ;
                         0415            ;
                         0415            ;      Schedule the message for transmission.
                         0415            ;
                         0415            ;
        1C A5   10   AB  0415            BISW    #XWB$M_FLG_SLI,XWB$W_FLG(R5)        ; We've got a message to send
  1C A5    4000 8F   AA  0419            BICW    #XWB$M_FLG_SDFL,XWB$Q_FLG(R5)       ; Whatever has just been built
                         041F                                                        ; satisfies the need to send the
                         041F                                                        ; backround inactivity message
                         041F      90$:  ;
                         041F            ;
                         041F            ;      Unless there are any wait conditions pending, solicit permission
                         041F            ;      from the Routing layer to transmit a message.
                         041F            ;
                         041F            ;
           0A   00   EA  041F            FFS     #0,#XWB$V_FLG_CLO+1,-               ; Get work bit
        50   1C A5       0422                       XWB$W_FLG(R5),R0                 ;
  0E FDBA CF    50   E1  0425            BBC      R0,NET$G[_WORKBITS,200$            ; Br if no work to be done
    08 0E A5    02   E2  0428            BBSS    #XWB$V_STS_SOL,XWB$W_STS(R5),100$; If BS, fork block in use
                         0430                                                        ;
                55   DD  0430            PUSHL    R5                                 ; Save XWB address
             FBCB'   30  0432            BSBW     NSP$SOLICIT                        ; Get permission to transmit
                55 8ED0  0435            POPL     R5                                 ; Restore XWB address
                     05  0438      100$: RSB
                         0439
    FA 0E A5    03   E1  0439      200$: BBC     #XWB$V_STS_DIS,XWB$W_STS(R5),100$; If BC, disconnect not pending
              0381   30  043E            BSBW     NET$CHK_X_IDLE                     ; Is XWB ready for disconnect ?
             F4 50   E9  0441            BLBC     R0,100$                            ; If LBC then no
              FEA5   31  0444            BRW      NET$FORK                           ; Attempt to resume disconnect
                         0447
```

```
            0447
            0447  ACT_DISPATCH:                               ; Dispatch action routine
            0447
            0447           $DISPATCH      TYPE=B,R4,                   -; R4 contains the action index
            0447           <-
            0447             <ACT$_ABORT,        ACT$ABORT>,           -;
            0447             <ACT$_BUG,          ACT$BUG>,             -;
            0447             <ACT$_CANLNK,       ACT$CANLNK>,          -;
            0447             <ACT$_CONFIRM,      ACT$CONFIRM>,         -;
            0447             <ACT$_DEACCESS,     ACT$DEACCESS>,        -;
            0447             <ACT$_ENT_RUN,      ACT$ENT_RUN>,         -;
            0447             <ACT$_INITIATE,     ACT$INITIATE>,        -;
            0447             <ACT$_LOG,          ACT$LOG>,             -;
            0447             <ACT$_NOP,          ACT$NOP>,             -;
            0447             <ACT$_RES_DISC,     ACT$RES_DISC>,        -;
            0447             <ACT$_RCV_CA,       ACT$RCV_CA>,          -;
            0447             <ACT$_RCV_CC,       ACT$RCV_CC>,          -;
            0447             <ACT$_RCV_CI,       ACT$RCV_CI>,          -;
            0447             <ACT$_RCV_CR,       ACT$RCV_CR>,          -;
            0447             <ACT$_RCV_DATA,     ACT$RCV_DATA>,        -;
            0447             <ACT$_RCV_DTACK,    ACT$RCV_DTACK>,       -;
            0447             <ACT$_RCV_DX        ACT$RCV_DX>,          -;
            0447             <ACT$_RCV_LI,       ACT$RCV_LI>,          -;
            0447             <ACT$_RCV_LIACK,    ACT$RCV_LIACK>,       -;
            0447             <ACT$_RCV_RTS,      ACT$RCV_RTS>,         -;
            0447             <ACT$_RTS_NLT,      ACT$RTS_NLT>,         -;
            0447             <ACT$_SHR[NK,       ACT$SHR[NK>,          -;
            0447             <ACT$_SSABORT,      ACT$SSABORT>,         -;
            0447           >
01   11     0477           BRB     ACT$BUG                     ; If unknown, bug
            0479
```

```
                             0479  .SBTTL  ACT$NOP         - Null action routine
                             0479  .SBTTL  ACT$BUG         - BUG_CHECK action routine
                             0479  .SBTTL  ACT$LOG         - Log-event action routine
                             0479  .SBTTL  ACT$NOLINK      - Report "SS$_FILNOTACC"
                             0479  .SBTTL  ACT$SSABORT     - Abort QIO since link was disconnected
                             0479
                         05  0479  ACT$NOP:        RSB
                             047A  ACT$BUG:        BUG_CHECK NETNOSTATE,FATAL
                         01  047E  ACT$LOG:        NOP
                         01  047F                  NOP
                             0480
                         01  0480                  NOP
                         01  0481                  NOP
                         01  0482                  NOP
                         05  0483                  RSB
                             0484
                             0484
 38 A3    00AC 8F        3C  0484  ACT$NOLINK:     MOVZWL  #SS$_FILNOTACC,IRP$L_IOST1(R3)
                         05  048A                  RSB
                             048B
                             048B  ACT$SHRLNK:     ;&nyi
      38 A3    2C        3C  048B  ACT$SSABORT:    MOVZWL  #SS$_ABORT,IRP$L_IOST1(R3)
                         05  048F                  RSB
                             0490
                             0490
                         0490  .SBTTL  NET$STARTIO     - Start I/O operation
                         0490  ;+
                         0490  ;
                         0490  ;  This routine is entered when the associated unit is idle and a packet
                         0490  ;  is available for processing.  The IRP$L_WIND field is used to locate the
                         0490  ;  associated XWB.
                         0490  ;
                         0490  ;
                         0490  ;  INPUTS:        R5      UCB address
                         0490  ;                 R4      PCB address
                         0490  ;                 R3      IRP address
                         0490  ;
                         0490  ;  OUTPUTS:        *** TBS ***
                         0490  ;
                         0490  ;-
                         0490  NET$STARTIO:                                            ; Process next IRP
        0FE0 8F     BB   0490           PUSHR    #^M<R5,R6,R7,R8,R9,R10,R11>     ; Setup "event" context
             44     10   0494           BSBB     PROC_IO                         ; Process the I/O function
          05 55     E8   0496           BLBS     R5,20$                          ; If LBS, no event to process
             5B     D4   0499           CLRL     R11                             ; Say "can't go to IPL 2"
           FEB8     30   049B           BSBW     NET$EVENT                       ; Process event in R7
        0FE0 8F     BA   049E  20$:     POPR     #^M<R5,R6,R7,R8,R9,R10,R11>     ; Return to UCB 'fork' context
                         04A2                                                    ;
     53    58 A5    D0   04A2           MOVL     UCB$L_IRP(R5),R3                ; Get IRP
             1F     13   04A6           BEQL     50$                             ; If EQL then its been queued
                         04A8                                                    ; or suspended, start next I/O
                         04A8           ;
                         04A8           ;
                         04A8           ;       Deallocate misc. buffer
                         04A8           ;
                         04A8           ;
   0C 2A A3   03    E1   04A8           BBC      #IRP$V_COMPLX,IRP$W_STS(R3),40$ ; If BC, IRP$L_DIAGBUF does not
                         04AD                                                    ; point to a NETDRIVER buffer
     50    4C A3    D0   04AD           MOVL     IRP$L_DIAGBUF(R3),R0            ; Get buffer
             06     18   04B1           BGEQ     40$                             ; If GEQ then none
          4C A3     D4   04B3           CLRL     IRP$L_DIAGBUF(R3)               ; Detach it
           0895     30   04B6           BSBW     NET$DEALLOCATE                  ; Deallocate block in R0
                         04B9  40$:     ;
                         04B9           ;
                         04B9           ;       Start next I/O.
                         04B9           ;
                         04B9           ;
     50    38 A3    D0   04B9           MOVL     IRP$L_IOST1(R3),R0              ; First  IOSB longword
     51    44 A5    D0   04BD           MOVL     UCB$L_DEVDEPEND(R5),R1          ; Second IOSB longword
                         04C1           REQCOM                                   ; Complete I/O, start next IRP
     53    4C B5    0F   04C7  50$:     REMQUE   @UCB$L_IOQFL(R5),R3             ; Get next IRP
             06     1D   04CB           BVS      60$                             ; If VS then none
    00000000'GF     17   04CD           JMP      G^IOC$INITIATE                  ; Deliver IRP to driver
 64 A5   0100 8F    AA   04D3  60$:     BICW     #UCB$M_BSY,UCB$W_STS(R5)        ; Free up the UCB
                    05   04D9           RSB                                      ; Return to Exec
                         04DA
                         04DA  PROC_IO:;
                         04DA           ;
                         04DA           ;       Move the UCB to R6 and the XWB (if any) to R5 and dispatch
                         04DA           ;       on function code with:
                         04DA           ;
                         04DA           ;          R10-R7  Scratch
```

```
                              04DA            ;       R6      UCB address
                              04DA            ;       R5      XWB address if LBC, else garbage
                              04DA            ;       R3      IRP address
                              04DA            ;       R2-R0   Scratch
                              04DA            ;
                              04DA            ;
                 56    55  D0 04DA            MOVL    R5,R6                             ; Copy UCB to safe register
            55    18   A3  D0 04DD            MOVL    IRP$L_WIND(R3),R5                 ; Get XWB, if any
                       03  19 04E1            BLSS    10$                               ; If LSS, XWB is in system space
                 55    01  C8 04E3            BISL    #1,R5                             ; Else, invalidate window ptr
        38 A3     00F4 8F  3C 04E6     10$:   MOVZWL  #SS$_ILLIOFUNC,IRP$L_IOST1(R3)    ; Assume fct not supported
57   20 A3   FFFFFFC0 8F   CB 04EC            BICL3   #^C<IO$M_FCODE>,IRP$W_FUNC(R3),R7 ; Get code without modifiers
                              04F5
                              04F5            $DISPATCH  R7,TYPE=B,-                    ; Process I/O
                              04F5            <-
                              04F5              <IO$_ACCESS,     NET$ACCESS>,-          ; Connect Requests
                              04F5              <IO$_DEACCESS,   NET$DEACCESS>,-        ; Disconnect Requests
                              04F5              <IO$_ACPCONTROL, NET$CONTROL>,-         ; ACP Control function
                              04F5            >                                         ; Else, fall thru
                 55    01  D0 0507            MOVL    #1,R5                             ; Set low bit to prevent event
                           05 050A            RSB                                       ; dispatching and return
                              050B
                              050B  .SBTTL  NET$FDT_SETMODE - Process IO$_SETMODE request
                              050B  ;+
                              050B  ;
                              050B  ; *** tbs ***
                              050B  ;
                              050B  ;-
                              050B  NET$FDT_SETMODE:                                      ; Process IO$_SETMODE function
            51    6C     D0   050B          MOVL    P1(AP),R1                             ; Get characteristics buffer
                  0B     13   050E          BEQL    10$                                   ; If EQL then none
                              0510          IFNORD  #8,(R1),50$                           ; Br on access violation
       44 A5    04 A1    D0   0516          MOVL    4(R1),UCB$L_DEVDEPEND(R5)             ; Set mailbox mask
    50    18 A3   01     CB   051B  10$:    BICL3   #1,IRP$L_WIND(R3),R0                  ; Get XWB address
                  00     18   0520          BGEQ    40$                                   ; If GEQ then none
                              0522          ;
                              0522          ;
                              0522          ;       This was used for "maintenance" mode.  Now available for
                              0522          ;       future functions.
                              0522          ;
                              0522          ;
           51    44 A5   D0   0522  40$:    MOVL    UCB$L_DEVDEPEND(R5),R1                ; Get device dependent info
              50    01   3C   0526          MOVZWL  S^#SS$_NORMAL,R0                      ; Setup I/O status
          00000000'GF    17   0529          JMP     G^EXE$FINISHIO                        ; Return success
                              052F
              50    0C   3C   052F  50$:    MOVZWL  #SS$_ACCVIO,R0                        ; Setup I/O status
          00000000'GF    17   0532          JMP     G^EXE$ABORTIO                         ; Abort I/O
                              0538
```

```
                           0538  .SBTTL  NET$FDT_CONTROL - IO$_ACPCONTROL FDT processing
                           0538  .SBTTL  NET$CONTROL     - IO$_ACPCONTROL "startio" processing
                           0538  ;+
                           0538  ;
                           0538  ;  The FDT routine simply routes the IRP through the Exec to the ACP.  The Exec
                           0538  ;  builds a "complex buffer" describing the control function.  The ACP will
                           0538  ;  requeue any IRP to the driver if it does not recognize the control function.
                           0538  ;  The driver has been designed to handle some of its own control functions
                           0538  ;  since many are protocol or control block format specific.
                           0538  ;
                           0538  ;
                           0538  ;  INPUTS:        R5      UCB Address
                           0538  ;                 R4      PCB Address
                           0538  ;                 R3      IRP Address
                           0538  ;
                           0538  ;  OUTPUTS:       R5      Unchanged
                           0538  ;                 R0      I/O status
                           0538  ;
                           0538  ;-
                           0538  NET$FDT_CONTROL:                                        ; FDT phase for IO$_ACPCONTROL
         4C A3      D4     0538          CLRL    IRP$L_DIAGBUF(R3)                      ; Zero misc buffer pointer
      18 A3   01    CA     053B          BICL    #1,IRP$L_WIND(R3)                      ; Always clear interlock flag
                           053F          ASSUME  PHD$Q_PRIVMSK  EQ  0                   ;
   40 A3   6C B4    7D     053F          MOVQ    @PCB$L_PHD(R4),IRP$Q_NT_PRVMSK(R3)     ; Store privilege mask
     00000000'GF    17     0544          JMP     G^ACP$MODIFY                           ; Continue in EXEC
                           054A
                           054A
                           054A  ;
                           054A  ;  INPUTS:        *** tbs ***
                           054A  ;  OUTPUTS:       *** tbs ***
                           054A  ;
                           054A  ;
                           054A  NET$CONTROL:                                            ; "Startio" for IO$_ACPCONTROL
   08 2A A3   03    E1     054A          BBC     #IRP$V_COMPLX,IRP$W_STS(R3),10$        ; If BC, part of $CANCEL
      50   2C B3    D0     054F          MOVL    @IRP$L_SVAPTE(R3),R0                   ; Get ptr to window descriptor
              60    D4     0553          CLRL    (R0)                                   ; Don't affect window upon
              1F    11     0555          BRB     50$                                    ;  I/O completion
                           0557  10$:    ;
                           0557          ;
                           0557          ;    The user is getting ready to issue an IO$_DEACCESS QIO to break the
                           0557          ;    link.  In order for the IO$_DEACCESS to be sent to the driver, the
                           0557          ;    channel's outstanding I/O count (CCB$W_IOC) must be zero.  Hence the
                           0557          ;    receiver must be run-down and any outstanding receive IRP's aborted.
                           0557          ;
                           0557          ;
      38 A3   01    3C     0557          MOVZWL  #SS$_NORMAL,IRP$L_IOST1(R3)       ; Set I/O status
         55   01    CA     055B          BICL    #1,R5                             ; Clear interlock bit
              2F    19     055E          BLSS    70$                               ; If LSS then valid XWB
                           0560          ;
                           0560          ;
                           0560          ;     Scan LTB to find XWB with an access pending for this channel
                           0560          ;
                           0560          ;
      50   34 A6    D0     0560          MOVL    UCB$L_VCB(R6),R0                  ; Get RCB
              10    13     0564          BEQL    50$                               ; If EQL then none
      50   24 A0    D0     0566          MOVL    RCB$L_PTR_LTB(R0),R0              ; Get LTB
              0A    13     056A          BEQL    50$                               ; If EQL then none
```

```
        55    E0 A0    9E   056C   1379        MOVAB   -XWB$L_LINK+LTB$L_XWB(R0),R5          ; Prepare for XWB scan
        55    2C A5    D0   0570   1380 20$:   MOVL     XWB$L_LINK(R5),R5                    ; Get next XWB
                 04    12   0574   1381        BNEQ     60$                                  ; If EQL then none left
           55    01    88   0576   1382 50$:   BISB     #1,R5                                ; Prevent event dispatching
                       05   0579   1383        RSB                                           ; Done
                            057A   1384
     50    0080 C5     D0   057A   1385 60$:   MOVL     XWB$L_IRP_ACC(R5),R0                 ; Get suspended IRP, if any
                 EF    13   057F   1386        BEQL     20$                                  ; If EQL none, loop
  0C A3    0C A0       D1   0581   1387        CMPL     IRP$L_PID(R0),IRP$L_PID(R3)          ; Belong to this process ?
                 E8    12   0586   1388        BNEQ     20$                                  ; Loop if NEQ
  28 A3    28 A0       B1   0588   1389        CMPW     IRP$W_CHAN(R0),IRP$W_CHAN(R3)        ; Same channel ?
                 E1    12   058D   1390        BNEQ     20$                                  ; If NEQ, loop
                            058F   1391 70$:   ;
                            058F   1392        ;
                            058F   1393        ;       The transmitter is not automatically run-down since the user may be
                            058F   1394        ;       preparing a "synchronous" disconnect -- i.e., disconnect after the
                            058F   1395        ;       final data segment has been ACK'd.  The manner in which pipelining
                            058F   1396        ;       has been implemented allows user transmit IRP's to be sent to I/O
                            058F   1397        ;       completion before the corresponding CXB's have been ACK'd  (or even
                            058F   1398        ;       sent).  Therefore, the user might issue a call to $CANCEL mistakenly
                            058F   1399        ;       thinking that the final message has be ACK'd.  Hence $CANCEL should
                            058F   1400        ;       allow the transmit CXB's to be ACK'd in their normal fashion.
                            058F   1401        ;
                            058F   1402        ;       Therefore, drain the receiver of all IRP's and CXB's.  If there
                            058F   1403        ;       are any transmit IRP's on the queue, then the disconnect is not
                            058F   1404        ;       synchronous, and the transmitter queue must be drained as well.
                            058F   1405        ;
                            058F   1406        ;
  0B 0E A5    04       E0   058F   1407        BBS      #XWB$V_STS_CON,XWB$W_STS(R5),80$ ; If BS, IO$_ACCESS pending
            038A       30   0594   1408        BSBW      DRAIN_RCV                          ; Drain the receiver
              50       D4   0597   1409        CLRL      R0                                 ; Init R0 for CHK_X_IRP call
            0235       30   0599   1410        BSBW      CHK_X_IRP                          ; Any Xmt IRP's
           D7 50       E8   059C   1411        BLBS      R0,50$                             ; If LBS, no
        57    0D       3C   059F   1412 80$:   MOVZWL   #NETEVT$_CANLNK,R7                   ; Force link to break
                       05   05A2   1413        RSB                                           ; Done
                            05A3   1414
                            05A3   1415
```

```
                         05A3 .SBTTL  NET$FDT_ACCESS  - IO$_ACCESS  FDT processing
                         05A3 .SBTTL  NET$ACCESS      - IO$_ACCESS "startio" processing
                         05A3 ;++
                         05A3 ;
                         05A3 ;   NET$FDT_ACCESS passes the IRP through the EXEC, where the user parameters
                         05A3 ;   are packaged into a "complex buffer", to the ACP.  The ACP reads the user
                         05A3 ;   connect info to build an Internal Connect Block (ICB) which it attaches to
                         05A3 ;   the IRP$L_DIAGBUF field of the IRP and requeues the IRP to the driver.  The
                         05A3 ;   role of the ACP is to lookup default access control (username, password,
                         05A3 ;   account) information in its data base and to translate node and object names
                         05A3 ;   to numbers.
                         05A3 ;
                         05A3 ;   NET$ACCESS reads the ICB and determines the type of connect.  It builds an
                         05A3 ;   XWB for connect initiate events and locates an already existing XWB for all
                         05A3 ;   others.  NET$ACCESS stores the appropriate event code in R7 and returns
                         05A3 ;   expecting the caller to call the event dispatcher.
                         05A3 ;
                         05A3 ;   It should be noted that the size of the XWB is not charged against the user
                         05A3 ;   byte count or byte limit quotas.  It is assumed that these quotas are at
                         05A3 ;   least partly used to limit a run away process and that the file quota of a
                         05A3 ;   process, against which logical-links are charged, is a sufficient mechanism.
                         05A3 ;
                         05A3 ;
                         05A3 ;
                         05A3 ;   INPUTS:      *** tbs ***
                         05A3 ;
                         05A3 ;   OUTPUT:      *** tbs ***
                         05A3 ;
                         05A3 ;
                         05A3 ;--
                         05A3 NET$FDT_ACCESS:                                     ; IO$_ACCESS  "FDT" processing
                         05A3
                         05A3         ASSUME  PHD$Q_PRIVMSK EQ 0
                         05A3
40 A3   6C B4      7D    05A3         MOVQ    @PCB$L_PHD(R4),IRP$Q_NT_PRVMSK(R3) ; Store priv mask in IRP
        4C A3      D4    05A8         CLRL    IRP$L_DIAGBUF(R3)                   ; Indicate no ICB
  00000000'GF      17    05AB         JMP     G^ACP$ACCESSNET                     ; Continue in EXEC
                         05B1
                         05B1
                         05B1 NET$ACCESS:                                         ; IO$_ACCESS "startio" processing
         023A      30    05B1         BSBW    GET_WNDSC                           ; Get CCB$L_WIND image descr.
           67      D4    05B4         CLRL    (R7)                                ; Init CCB$L_WIND image
2A A3      02      A8    05B6         BISW    #IRP$M_FUNC,IRP$W_STS(R3)           ; Mark for write back
32 A3      01      3C    05BA         MOVZWL  #1,IRP$L_BCNT(R3)                   ; Write back one descriptor
   55      56      D0    05BE         MOVL    R6,R5                               ; Copy UCB addr for subroutines
   58      53      D0    05C1         MOVL    R3,R8                               ; Copy IRP address to safe reg
   54   4C A8      D0    05C4         MOVL    IRP$L_DIAGBUF(R8),R4                ; Get ICB pointer
           3C      18    05C8         BGEQ    80$                                 ; If GEQ, its an error code
                         05CA 10$:    ;
                         05CA         ;
                         05CA         ;       IO$_ACCESS made it successfully through the ACP
                         05CA         ;
                         05CA         ;
   53   02 A4      3C    05CA         MOVZWL  ICB$W_LOCLNK(R4),R3                 ; Get local link address
         06C6      30    05CE         BSBW    XWB_LOCLNK                          ; Find associated XWB
        2D 55      E8    05D1         BLBS    R5,60$                              ; Br if XWB was not found
0C A8   34 A5      D1    05D4         CMPL    XWB$L_PID(R5),IRP$L_PID(R8)         ; PIDs match ?
```

```
             1F  12  05D9  1474            BNEQ    55$                             ; Br if they don't
          3C A5  B5  05DB  1475            TSTW    XWB$W_REMLNK(R5)                ; Does remote link id exit ?
             0B  12  05DE  1476            BNEQ    30$                             ; Connect Confirm if NEQ
                     05E0  1477            ;
                     05E0  1478            ;
                     05E0  1479            ;       Connect Initiate
                     05E0  1480            ;
                     05E0  1481            ;
       04 A4 01  A1  05E0  1482            ADDW3   #1,ICB$W_TIM_OCON(R4),-         ; Setup outbound connect timer
          50 A5      05E4  1483                    XWB$W_TIMER(R5)                 ; (+1 for possible clock skew)
          57  0F 9A  05E6  1484            MOVZBL  #NETEVT$_CIA,R7                 ; Set 'connect initiate access'
              0B 11  05E9  1485            BRB     50$                             ; Finish in common
                     05EB  1486 30$:       ;
                     05EB  1487            ;
                     05EB  1488            ;       Connect Confirm
                     05EB  1489            ;
                     05EB  1490            ;
          57  10 9A  05EB  1491 40$:       MOVZBL  #NETEVT$_CCA,R7                 ; Set 'connect confirm access'
    03 20 A8  08 E1  05EE  1492            BBC     #IO$V_ABORT,IRP$W_FUNC(R8),50$  ; If BC, not Connect Reject
          57  11 9A  05F3  1493            MOVZBL  #NETEVT$_CRA,R7                 ; Set 'connect reject access'
                     05F6  1494 50$:       ;
                     05F6  1495            ;
                     05F6  1496            ;       Because the low bit of R5 is clear, the XWB will considered to
                     05F6  1497            ;       be valid and the event in R7 will be processed.
                     05F6  1498            ;
                     05F6  1499            ;
          53  58 D0  05F6  1500            MOVL    R8,R3                           ; Setup IRP address
              05     05F9  1501            RSB                                     ; Return with LBC in R5
                     05FA  1502
                     05FA  1503 ;
                     05FA  1504 ;  Unsuccessful access
                     05FA  1505 ;
   54   0840 8F  3C  05FA  1506 55$:       MOVZWL  #SS$_DEVALLOC,R4                ; Setup error code
              05 11  05FF  1507            BRB     80$                             ; Continue
   54   20DC 8F  3C  0601  1508 60$:       MOVZWL  #SS$_CONNECFAIL,R4              ; Setup error code
          53  58 D0  0606  1509 80$:       MOVL    R8,R3                           ; Setup IRP pointer
       38 A3  54 D0  0609  1510            MOVL    R4,IRP$L_IOST1(R3)              ; Store error code
          55  01 D0  060D  1511            MOVL    #1,R5                           ; Tell CLEANUP_ACCESS "no XWB"
            0201 30  0610  1512            BSBW    CLEANUP_ACCESS                  ; Restore quota
              05     0613  1513            RSB                                     ; On return goto REQCOM
                     0614  1514
                     0614  1515 100$:      BUG_CHECK  NETNOSTATE,FATAL
                     0618  1516
```

```
                        0618  .SBTTL  ACT$INITIATE    - Connect Initiate action routine
                        0618  .SBTTL  ACT$CONFRIM     - Connect Confirm action routine
                        0618  ;+
                        0618  ;
                        0618  ;  These action routines resume processing the event setup by NET$ACCESS.
                        0618  ;  ACT$INITIATE assumes that a Connect Initiate message will be built
                        0618  ;  and sent.  ACT$CONFIRM is used when a received connect is being either
                        0618  ;  accepted or rejected and assumes that eihter a Connect Confirm or a
                        0618  ;  Disconnect Initiate message will be built and sent.
                        0618  ;
                        0618  ;
                        0618  ;  INPUTS:        R8       Scratch
                        0618  ;                R7       Event code
                        0618  ;                R6       UCB address
                        0618  ;                R5       XWB address
                        0618  ;                R4       Scratch
                        0618  ;                R3       IRP address
                        0618  ;                R2-R0    Scratch
                        0618  ;
                        0618  ;  OUTPUTS:       R8,R7    Garbage
                        0618  ;                R6,R5    Preserved
                        0618  ;                R4-R0    Garbage
                        0618  ;
                        0618  ;-
                        0618  ACT$CONFIRM::                                               ; Connect Confirm or Reject
            35    10    0618          BSBB    SETUP_XWB                                   ; Do common setup
                        061A          ;
                        061A          ;
                        061A          ;       If the remote end of the Logical-link is on the local node then
                        061A          ;       use the same "path".  This allows loopbacked lines to be used for
                        061A          ;       all logical-link traffic in both directions -- which seems like a
                        061A          ;       sensible thing to do even though this may be a depature from the
                        061A          ;       Network Management architecture.
                        061A          ;
                        061A          ;
         38 A5    B5    061A          TSTW    XWB$W_PATH(R5)                              ; Has a path been chosen ?
            21    12    061D          BNEQ    20$                                         ; If NEQ then yes
    52   30 A5    D0    061F          MOVL    XWB$L_VCB(R5),R2                            ; Get the RCB
 0E A2   3A A5    B1    0623          CMPW    XWB$W_REMNOD(R5),RCB$W_ADDR(R2)             ; Is the remote node us?
            16    12    0628          BNEQ    20$                                         ; If NEQ no
                        062A                                                              ;
            38    BB    062A          PUSHR   #^M<R3,R4,R5>                               ; Save regs
    53   3C A5    3C    062C          MOVZWL  XWB$W_REMLNK(R5),R3                         ; Get remote link i.d.
          0676    30    0630          BSBW    NET$XWB_LOCLNK                              ; Find corresponding XWB
      52    55    D0    0633          MOVL    R5,R2                                       ; Copy XWB address to new reg
            38    BA    0636          POPR    #^M<R3,R4,R5>                               ; Restore regs
                        0638                                                              ;
         05 52    E8    0638          BLBS    R2,20$                                      ; If LBS then no XWB was found
 38 A5   38 A2    B0    063B          MOVW    XWB$W_PATH(R2),XWB$W_PATH(R5)               ; Use partner's path i.d.
09 20 A3    08    E1    0640  20$:    BBC     #IO$V_ABORT,IRP$W_FUNC(R3),100$             ; If BC then not connect reject
                        0645          ASSUME  NET$C_DR_NORMAL  EQ  0                      ;
         46 A5    B4    0645          CLRW    XWB$W_X_REASON(R5)                          ; Setup disconnect reason code
      50    01    7D    0648          MOVQ    S^#SS$_NORMAL,R0                            ; Setup IOSB value
          007C    30    064B          BSBW    NET$CMPL_ACC                                ; Complete the IO$_ACCESS IRP
                  05    064E  100$:   RSB                                                 ; Done
                        064F
                        064F  ACT$INITIATE::                                              ; Connect Initiate request
```

```
                            064F  1575 SETUP_XWB:                                            ; Setup common fields
            0C A5     B6    064F  1576         INCW    XWB$W_REFCNT(R5)                      ; Account for accessor
  016C C5   34 A5     D0    0652  1577         MOVL    XWB$L_PID(R5),XWB$$+ACB$L_PID(R5) ; Setup Special Kernal AST PID
     0080 C5   53     D0    0658  1578         MOVL    R3,XWB$L_IRP_ACC(R5)                  ; Store IRP address
            58 A6     D4    065D  1579         CLRL    UCB$L_IRP(R6)                         ; Clear IRP pointer to prevent
                            0660  1580                                                       ; immediate I/O completion
      10 A5    56     D0    0660  1581         MOVL    R6,XWB$L_ORGUCB(R5)                   ; Setup UCB ptr
        54   4C A3    D0    0664  1582         MOVL    IRP$L_DIAGBUF(R3),R4                  ; Get ICB ptr
             4C A3    D4    0668  1583         CLRL    IRP$L_DIAGBUF(R3)                     ; Dett ch it from IRP
     010C C5   54     D0    066B  1584         MOVL    R4,XWB$L_ICB(R5)                      ; Attach it to XWB
  1C A5   0100 8F     A8    0670  1585         BISW    #XWB$M_FLG_SCD,XWB$W_FLG(R5)          ; Set send message flag
                            0676  1586 30$:    ;
                            0676  1587         ;
                            0676  1588         ;       Setup pre-allocated byte quota to take upon entering the RUN state
                            0676  1589         ;
                            0676  1590         ;
                            0676  1591 ;&      CLRW    XWB$W_X_QUO(R5)                       ; Pre-allocate none for rcv's
                            0676  1592 ;&      CLRW    XWB$W_R_QUO(R5)                       ; Pre-allocate none for rcv's
                            0676  1593         ;
                            0676  1594         ;
                            0676  1595         ;       Move remainder of parameters from the ICB
                            0676  1596         ;
                            0676  1597         ;
                 38   BB    0676  1598         PUSHR   #^M<R3,R4,R5>                                 ; Save MOVC regs
                            0678  1599                                                               ;
        51   7C A4    9E    0678  1600         MOVAB   ICB$B_DATA(R4),R1                             ; Get source pointer
             06E0     30    067C  1601         BSBW    NET$MOV_TO_XWB                                ; Move counted string
                            067F  1602                                                               ; to XWB$B_DATA...
                            067F  1603         ASSUME  ICB$C_RID  LE  XWB$C_RID                      ;
  6F A5   0092 C4     90    067F  1604         MOVB    ICB$B_RID(R4),XWB$B_RID(R5)                   ; Move the count field
                            0685  1605                                                               ;
20   0093 C4   10     2C    0685  1606         MOVC5   #ICB$C_RID,ICB$T_RID(R4),#^A' ',-             ; Move the remote
        70 A5  10           068B  1607                 #XWB$C_RID,XWB$T_RID(R5)                      ; i.d. text
                            068E  1608                                                               ;
                 38   BA    068E  1609         POPR    #^M<R3,R4,R5>                                 ; Restore regs
        50   0C A4    B0    0690  1610         MOVW    ICB$W_RETRAN(R4),R0                           ; Get rexmt factor
                 04   15    0694  1611         BLEQ    50$                                           ; If LEQ keep default
      54 A5    50     B0    0696  1612         MOVW    R0,                XWB$W_RETRAN(R5)           ; Update rexmt factor
            52 A5     B4    069A  1613 50$:    CLRW                       XWB$W_PROGRESS(R5)         ; Init progress count
      38 A5    64     B0    069D  1614         MOVW    ICB$W_PATH(R4),    XWB$W_PATH(R5)             ; Circuit to use
    40 A5   12 A4     B0    06A1  1615         MOVW    ICB$W_SEGSIZ(R4),  XWB$W_LOCSIZ(R5)           ; Rcv buffer size
    56 A5   0E A4     B0    06A6  1616         MOVW    ICB$W_DLY_FACT(R4), XWB$W_DLY_FACT(R5)        ; Delay factor
    58 A5   10 A4     B0    06AB  1617         MOVW    ICB$W_DLY_WGHT(R4), XWB$W_DLY_WGHT(R5)        ; Delay weight
    4C A5   06 A4     B0    06B0  1618         MOVW    ICB$W_TIM_INACT(R4),XWB$W_TIM_INACT(R5)       ; Inactivity timer
                            06B5  1619         ;
                            06B5  1620         ;
                            06B5  1621         ;       Setup TIMER and DELAY so that the Connect message will be
                            06B5  1622         ;       retransmitted periodically if necessary.  This is done by choosing
                            06B5  1623         ;       a DELAY which will allow RETRAN retransmission before the amount
                            06B5  1624         ;       of time left in TIMER expires.
                            06B5  1625         ;
                            06B5  1626         ;
            54 A5     A7    06B5  1627         DIVW3   XWB$W_RETRAN(R5),-                    ; TIMER has number of ticks
    4E A5   50 A5           06B8  1628                 XWB$W_TIMER(R5),XWB$W_DELAY(R5)       ; left before timeout
            56 A5     A6    06BC  1629         DIVW    XWB$W_DLY_FACT(R5),-                  ; Adjust for the "delay factor"
            4E A5           06BF  1630                 XWB$W_DELAY(R5)                       ;
               03     12    06C1  1631         BNEQ    70$                                   ; If NEQ then use it
      4E A5  B6  06C3  1632          INCW    XWB$W_DELAY(R5)                ; Else use 1 second
      F937'  30  06C6  1633 70$:     BSBW    NET$RESET_TIMER                ; Reset XWB$W_TIMER
             05  06C9  1634          RSB                                    ; Done
                 06CA  1635
```

```
                   06CA  .SBTTL  NET$CMPL_ACC    - Complete IO$_ACCESS, fill in window
                   06CA  ;++
                   06CA  ;
                   06CA  ;  The access function currently being processed is completed.
                   06CA  ;  If the I/O completion status is not successful then the window of the
                   06CA  ;  channel associated with the IRP is cleared.
                   06CA  ;
                   06CA  ;
                   06CA  ;  INPUTS:  R5  XWB address
                   06CA  ;           R1  Second IOSB longword value
                   06CA  ;           R0  First  IOSB longword value
                   06CA  ;
                   06CA  ;  OUTPUTS: R1  Garbage
                   06CA  ;           R0  SS$_NORMAL
                   06CA  ;
                   06CA  ;           All other registers are preserved.
                   06CA  ;
                   06CA  ;--
                   06CA  NET$CMPL_ACC::                                  ; Complete access, fill in window
      01BC 8F   BB 06CA          PUSHR   #^M<R2,R3,R4,R5,R7,R8>          ; Save regs
                   06CE                                                  ;
04   4E A5      B1 06CE          CMPW    XWB$W_DELAY(R5),#4              ; Make sure initial "delay" estimate
        04      1E 06D2          BGEQU   30$                             ; is at least 4 seconds
  4E A5 04      B0 06D4          MOVW    #4,XWB$W_DELAY(R5)              ;
53  0080 C5     D0 06D8  30$:    MOVL    XWB$L_IRP_ACC(R5),R3            ; Get IO$_ACCESS IRP
        24      13 06DD          BEQL    200$                            ; If EQL then none
     0080 C5    D4 06DF          CLRL    XWB$L_IRP_ACC(R5)               ; Remove IRP
  38 A3 50      7D 06E3          MOVQ    R0,IRP$L_IOST1(R3)              ; Save I/O status
                   06E7          ;
                   06E7          ;
                   06E7          ;    Setup the CCB$L_WIND value and deallocate the ICB.
                   06E7          ;
                   06E7          ;    If either the access was unsuccessful or the request was for a
                   06E7          ;    Connect Reject, then cleanup from the IO$_ACCESS attempt and
                   06E7          ;    leave the CCB$L_WIND image at zero.
                   06E7          ;
                   06E7          ;
        08      E0 06E7          BBS     #IO$V_ABORT,-                   ; If BS, Connect Reject
   06 20 A3        06E9                  IRP$W_FUNC(R5),60$              ;
     5B A5      94 06EC          CLRB    XWB$B_DATA(R5)                  ; Init optional data cell to prepare
                   06EF                                                  ; for eventual disconnect
     05 50      E8 06EF          BLBS    R0,100$                         ; If LBS then successful IO$_ACCESS
      011F      30 06F2  60$:    BSBW    CLEANUP_ACCESS                  ; Clean up from access I/O fct
        09      11 06F5          BRB     110$                            ; Complete the I/O
      00F4      30 06F7  100$:   BSBW    GET_WNDSC                       ; Get CCB$L_WIND image descriptor
   67   55      D0 06FA          MOVL    R5,(R7)                         ; Setup CCB$L_WIND value
      0161      30 06FD          BSBW    DEAL_ICB                        ; Deallocate the ICB
                   0700  110$:   ;
                   0700          ;
                   0700          ;    Complete the I/O
                   0700          ;
                   0700          ;
      067D      30 0700          BSBW    NET$POST_IO                     ; Post IRP for completion
                   0703                                                  ;
      01BC 8F   BA 0703  200$:   POPR    #^M<R2,R3,R4,R5,R7,R8>          ; Restore regs
   50   01      D0 0707          MOVL    S^#SS$_NORMAL,R0                ; Success
                05 070A          RSB
```

```
          070B  1694
          070B  1695
```

```
                     070B  .SBTTL  ACT$ENT_RUN     - Enter RUN state action routine
                     070B  ;+
                     070B  ;
                     070B  ;  This routine is entered to setup the XWB for entering the RUN state.
                     070B  ;
                     070B  ;  INPUTS:  R7  Event code  - it will be reprocessed via the complex event
                     070B  ;                             mechanism.  Note that the state should have
                     070B  ;                             been updated by then.
                     070B  ;           R5  XWB address
                     070B  ;           R0  Scratch
                     070B  ;
                     070B  ;  OUTPUTS: R0 garbage
                     070B  ;
                     070B  ;           All other registers are preserved.
                     070B  ;
                     070B  ;-
                     070B  ACT$ENT_RUN::                           ; Enter RUN state
      F8F2'   30     070B          BSBW    NET$SETUP_RUN           ; Setup XWB for RUN state
      FC1F    31     070E          BRW     NET$COMPLEX_EV          ; Change state and resignal the event
                     0711
```

```
                        0711  .SBTTL  NET$FDT_DEACCESS- IO$_DEACCESS FDT processing
                        0711  .SBTTL  NET$DEACCESS    - IO$_DEACCESS "startio" processing
                        0711  ;++
                        0711  ;
                        0711  ;  INPUTS:       AP      Pointer to the QIO P1 parameter
                        0711  ;                R8      Must be saved/restored if return to Exec for next
                        0711  ;                        FDT routine
                        0711  ;                R7      I/O function code without modifiers
                        0711  ;                R6      CCB address
                        0711  ;                R5      UCB address
                        0711  ;                R4      PCB address
                        0711  ;                R3      IRP address
                        0711  ;                R2-R0   Scratch
                        0711  ;
                        0711  ;  OUTPUTS:      R5,R3   Preserved
                        0711  ;
                        0711  ;                All other regs may be clobbered.
                        0711  ;
                        0711  ;--
                        0711  NET$FDT_DEACCESS::                                            ; IO$_DEACCESS FDT routine
          4C A3    D4   0711          CLRL    IRP$L_DIAGBUF(R3)                             ; Zero misc buffer pointer
  50   00AC 8F     3C   0714          MOVZWL  #SS$_FILNOTACC,R0                             ; Assume "link not accessed"
     18 A3   01    CA   0719          BICL    #1,IRP$L_WIND(R3)                             ; Clear interlock bit
             52    18   071D          BGEQ    200$                                          ; If GEQ, link is not accessed
                        071F                                                                ;
        0F78 8F    BB   071F          PUSHR   #^M<R3,R4,R5,R6,R8,R9,R10,R11>                ; Save regs
                        0723          DSBINT  UCB$B_FIPL(R5)                                ; Synchronize
          5B  01   D0   072A          MOVL    #1,R11                                        ; Say "okay to go to IPL 2"
                        072D                                                                ;
    55    18 A3    D0   072D          MOVL    IRP$L_WIND(R3),R5                             ; Switch to XWB context
                        0731          ;
                        0731          ;
                        0731          ;    Setup disconnect reasons codes as appropriate
                        0731          ;
                        0731          ;
 44 A5    0064 8F  B1   0731          CMPW    #NET$C_DR_INVALID,XWB$W_R_REASON(R5)  ; Rcv'd reason code yet ?
             06    12   0737          BNEQ    10$                                   ; If NEQ yes
 44 A5    0066 8F  3C   0739          MOVZWL  #NET$C_DR_DEACC,  XWB$W_R_REASON(R5)  ; Setup local reason
 46 A5    0064 8F  B1   073F  10$:    CMPW    #NET$C_DR_INVALID,XWB$W_X_REASON(R5)  ; Xmt reason code setup ?
             0D    1A   0745          BGTRU   20$                                   ; If GTRU, yes
     46 A5   00    3C   0747          MOVZWL  #NET$C_DR_NORMAL, XWB$W_X_REASON(R5)  ; Assume ordinary disconn.
  04 20 A3   08    E1   074B          BBC     #IO$V_ABORT,      IRP$W_FUNC(R3),20$  ; If BS, must abort all I/O
     46 A5   09    3C   0750          MOVZWL  #NET$C_DR_ABORT,  XWB$W_X_REASON(R5)  ; Else, set "disc. abort"
                        0754  20$:    ;
                        0754          ;
                        0754          ;   If IO$V_ABORT is set, then both the transmitter and receiver must
                        0754          ;   be run-down.  Otherwise, this is a "synchronous" disconnect and
                        0754          ;   the transmitter must be allowed to send all data before breaking
                        0754          ;   the link.
                        0754          ;
                        0754          ;
  08 0E A5   04    E0   0754          BBS     #XWB$V_STS_CON,   XWB$W_STS(R5),100$  ; If BS, not in RUN format
  03 20 A3   08    E1   0759          BBC     #IO$V_ABORT,      IRP$W_FUNC(R3),50$  ; If BS, must abort all I/O
           0161    30   075E          BSBW    DRAIN_XMT                             ; Run-down the transmitter
           01BD    30   0761  50$:    BSBW    DRAIN_RCV                             ; Run-down the receiver
                        0764                                                  ;
                        0764  100$:   ENBINT                                  ; Restore IPL
           0F78 8F   BA  0767          POPR    #^M<R3,R4,R5,R6,R8,R9,R10,R11>  ; Restore regs
                         076B                                                  ;
        00000000'GF  17  076B          JMP     G^ACP$DEACCESS                  ; Goto system FDT routine
        00000000'GF  17  0771  200$:   JMP     G^EXE$ABORTIO                   ; Abort the I/O
                         0777
                         0777
                         0777  NET$DEACCESS::                                  ; User QIO to break link
               0074  30  0777          BSBW    GET_WNDSC                       ; Get CCB$L_WIND image desc
                 67  D4  077A          CLRL    (R7)                            ; Clear CCB$L_WIND image
       38 A3     01  3C  077C          MOVZWL  #SS$_NORMAL,IRP$L_IOST1(R3)     ; Setup success status
       2A A3     02  A8  0780          BISW    #IRP$M_FUNC,IRP$W_STS(R3)       ; Mark for write back
       32 A3     01  D0  0784          MOVL    #1,IRP$L_BCNT(R3)               ; Write back 1 (the window) ABD
          57     12  D0  0788          MOVL    #NETEVT$_DEA,R7                 ; Setup event code in case R5
                 05      078B          RSB                                     ; is a valid XWB pointer
                         078C
                         078C
                         078C  ACT$DEACCESS::                                  ; User QIO to break link
               0085  30  078C          BSBW    CLEANUP_ACCESS                  ; Clean up from access I/O fct
               0065  30  078F          BSBW    GET_P2DSC                       ; Get optional data descriptor
                 58  D7  0792          DECL    R8                              ; Reduce length by count field
                 1C  19  0794          BLSS    ACT$RES_DISC                    ; If LSS, then no data
          51     67  9A  0796          MOVZBL  (R7),R1                         ; Get count value from string
          51     58  D1  0799          CMPL    R8,R1                           ; Take minimum of size from
                 03  1F  079C          BLSSU   20$                             ;  descriptor and size from
          58     51  D0  079E          MOVL    R1,R8                           ;  string
          10     58  D1  07A1  20$:    CMPL    R8,#16                          ; Take minimum of what's there
                 03  1F  07A4          BLSSU   30$                             ;  and max allowed by NSP
          58     10  D0  07A6          MOVL    #16,R8                          ;
          67     58  90  07A9  30$:    MOVB    R8,(R7)                         ; Setup count field in string
          51     57  D0  07AC          MOVL    R7,R1                           ; Setup source ptr
               05AD  30  07AF          BSBW    NET$MOV_TO_XWB                  ; Move counted string to
                         07B2                                                  ; XWB$B_DATA
                         07B2  ACT$RES_DISC::                                  ; Resume Disconnect processing
                         07B2          ;
                         07B2          ;
                         07B2          ;       If the XWB is idle, continue processing this event.  Else, dismiss
                         07B2          ;       this event for now and resume it when the XWB goes idle.  This is
                         07B2          ;       the only way to do a "synchronous disconnect" with NSP pipelining
                         07B2          ;       causing user Transmit IRP's to be completed before the CXB's are
                         07B2          ;       actually transmitted.
                         07B2          ;
                         07B2          ;
                 0E  10  07B2          BSBB    NET$CHK_X_IDLE                  ; Is the transmitter idle ?
              04 50  E9  07B4          BLBC    R0,100$                         ; If LBS then no
               00F5  30  07B7          BSBW    NET$PURG_RUN                    ; Cleanup if necessary
                 05      07BA          RSB                                     ; Return to change state
                         07BB                                                  ;
       0E A5     08  A8  07BB  100$:   BISW    #XWB$M_STS_DIS,XWB$W_STS(R5)    ; Mark disconnect pending
               FB68  31  07BF          BRW     NET$END_EVENT                   ; Dismiss this event for now
                         07C2
                         07C2   1826
                         07C2   1827                    .ENABL  LSB
                         07C2   1828
                         07C2   1829 NET$CHK_X_IDLE::                                       ; See if transmitter is idle
   20 0E A5    04    E0  07C2   1830          BBS       #XWB$V_STS_CON,XWB$W_STS(R5),10$;  If BS, not in RUN format
   50   00BC C5      D0  07C7   1831          MOVL       XWB$T_DT+[SB$L_X_CXB(R5),R0     ; Get next DATA CXB
   50   00EC C5      C8  07CC   1832          BISL       XWB$T_LI+LSB$L_X_CXB(R5),R0     ;  OR in next Interrupt CXB
                         07D1   1833 CHK_X_IRP:                                         ; Check for Xmt IRP's
   50   00B4 C5      C8  07D1   1834          BISL       XWB$T_DT+LSB$L_X_PND(R5),R0     ;  OR in pending DATA Xmt IRP's
   50   00B8 C5      C8  07D6   1835          BISL       XWB$T_DT+LSB$L_X_IRP(R5),R0     ;  OR in spent DATA Xmt IPR's
   50   00E4 C5      C8  07DB   1836          BISL       XWB$T_LI+LSB$L_X_PND(R5),R0     ;  OR in pending Int. Xmt IRP's
   50   00B8 C5      C8  07E0   1837          BISL       XWB$T_DT+LSB$L_X_IRP(R5),R0     ;  OR in spent Int. Xmt IPR's
               04    12  07E5   1838          BNEQ       20$                             ; If NEQ then not idle
          50   01    D0  07E7   1839 10$:     MOVL      #1,R0                            ; Say "idle"
               05        07EA   1840          RSB                                        ; Done
                         07EB   1841                                                     ;
               50    D4  07EB   1842 20$:     CLRL       R0                              ; Say "non-idle"
               05        07ED   1843          RSB                                        ; Done
                         07EE   1844
                         07EE   1845                    .DSABL  LSB
                         07EE   1846
                         07EE   1847
                         07EE   1848                    .ENABL  LSB
                         07EE   1849 GET_WNDSC:                                 ; Get window descriptor
               58    D4  07EE   1850          CLRL      R8                      ; Get descriptor offset
               12    11  07F0   1851          BRB       10$                     ; Continue
                         07F2   1852 GET_P1DSC:                                 ; Get P1 descriptor
          58   08    D0  07F2   1853          MOVL      #8,R8                   ; Get descriptor offset
               0D    11  07F5   1854          BRB       10$                     ; Continue
                         07F7   1855 GET_P2DSC:                                 ; Get P2 descriptor
          58   10    D0  07F7   1856          MOVL      #8*2,R8                 ; Get descriptor offset
               08    11  07FA   1857          BRB       10$                     ; Continue in common
                         07FC   1858 GET_P3DSC:                                 ; Get P3 descriptor
          58   18    D0  07FC   1859          MOVL      #8*3,R8                 ; Get descriptor offset
               03    11  07FF   1860          BRB       10$                     ; Continue in common
                         0801   1861 GET_P4DSC:                                 ; Get P4 descriptor
          58   20    D0  0801   1862          MOVL      #8*4,R8                 ; Get desciptor offset
     58   2C B3      C0  0804   1863 10$:     ADDL      @IRP$L_SVAPTE(R3),R8    ; Get descriptor address
          57   88    3C  0808   1864          MOVZWL    (R8)+,R7                ; Get offset to data
   57     FF A847    9E  080B   1865          MOVAB     -1(R8)[R7],R7           ; Get ptr to data after skipping
                         0810   1866                                            ; over access mode byte
          58   68    3C  0810   1867          MOVZWL    (R8),R8                 ; Get length of data
               05        0813   1868          RSB
                         0814   1869                    .DSABL  LSB
                         0814   1870
```

```
                       0814  .SBTTL  CLEANUP_ACCESS  - Cleanup XWB for terminated IO$_ACCESS
                       0814  ;+
                       0814  ;
                       0814  ;  INPUTS:       R5 - XWB address, low bit set if none
                       0814  ;                R3 - IRP address
                       0814  ;
                       0814  ;  OUTPUTS:      All registers are preserved.
                       0814  ;
                       0814  ;
                       0814  ;-
                       0814  CLEANUP_ACCESS:
             07    BB  0814          PUSHR   #^M<R0,R1,R2>                         ; Save regs
                       0816                                                        ;
             52    D4  0816          CLRL    R2                                    ; Assume no byte quota to return
          19 55    E8  0818          BLBS    R5,20$                                ; If LBS then no XWB
             44    10  081B          BSBB    DEAL_ICB                              ; Deallocate the ICB
           04F6    30  081D          BSBW    NET$DRAIN_FREE_CXB                    ; Drain CXB free queue
          0C A5    B7  0820          DECW    XWB$W_REFCNT(R5)                      ; Account for loss of accessor
             38    12  0823          BNEQ    200$                                  ; Br if last accessor
          10 A5    D4  0825  10$:    CLRL    XWB$L_ORGUCB(R5)                      ; XWB is unowned
          34 A5    D4  0828          CLRL    XWB$L_PID(R5)                         ; XWB is unowned
  04 0E A5   0A    E0  082B          BBS     #XWB$V_STS_ASTPND,XWB$W_STS(R5),20$ ; If BS, AST block in use
        016C C5    D4  0830          CLRL    XWB$$TACB$L_PID(R5)                   ; Prevent false AST delivery
                       0834  20$:    ;
                       0834          ;
                       0834          ;       Return BYTCNT and FILCNT quota
                       0834          ;
                       0834          ;
      50   0C A3   3C  0834          MOVZWL  IRP$L_PID(R3),R0                      ; Get PID index
  51   00000000'GF D0  0838          MOVL    G^SCH$GL_PCBVEC,R1                    ; Get PCB vector address
        51   6140  D0  083F          MOVL    (R1)[R0],R1                           ; Get PCB address
   0C A3   60 A1   D1  0843          CMPL    PCB$L_PID(R1),IRP$L_PID(R3)           ; Still there ?
             10    12  0848          BNEQ    30$                                   ; If not branch
     50   0080 C1  D0  084A          MOVL    PCB$L_JIB(R1),R0                      ; Get JIB
          30 A0    B6  084F          INCW    JIB$W_FILCNT(R0)                      ; Return quota for IO$_ACCESS
      20 A0  52    C0  0852          ADDL    R2,JIB$L_BYTCNT(R0)                   ; Return byte quota
      24 A0  52    C0  0856          ADDL    R2,JIB$L_BYTLM(R0)                    ; Here too
                       085A                                                        ;
             07    BA  085A  30$:    POPR    #^M<R0,R1,R2>                         ; Restore regs
             05        085C          RSB                                           ; Done
                       085D
                       085D  200$:   BUG_CHECK NETNOSTATE,FATAL                    ; Invalid reference count
                       0861
                       0861  DEAL_ICB:                                             ; Deallocate the ICB
       7E    50    7D  0861          MOVQ    R0,-(SP)                              ; Save regs
                       0864                                                        ;
   0E 0E A5  04    E1  0864          BBC     #XWB$V_STS_CON,XWB$W_STS(R5),40$ ; If BC, XWB$L_ICB is invalid
     50   010C C5  D0  0869          MOVL    XWB$L_ICB(R5),R0                      ; Get buffer for deallocation
             03    18  086E          BGEQ    30$                                   ; If GEQ then none
           04DB    30  0870          BSBW    NET$DEALLOCATE                        ; Deallocate block in R0
        010C C5    D4  0873  30$:    CLRL    XWB$L_ICB(R5)                         ; Say "no ICB"
                       0877                                                        ;
       50    8E    7D  0877  40$:    MOVQ    (SP)+,R0                              ; Restore regs
             05        087A          RSB                                           ; Done
                       087B
                       087B  .SBTTL  NET$CANCEL      - Cancel I/O routine
                       087B  ;+
                       087B  ;
                       087B  ;  Most of the work for the Cancel-I/O sequence will occur when the special
                       087B  ;  IO$_ACPCONTROL QIO is issued by the $CANCEL system service.
                       087B  ;
                       087B  ;  In all cases, the ACP is informed via a mailbox message since special
                       087B  ;  cleanup may be needed in the ACP (e.g. declared name cleanup).  Note that
                       087B  ;  the special Cancel IRP is only sent to the ACP if there is a logical-link
                       087B  ;  active.
                       087B  ;
                       087B  ;
                       087B  ;  INPUTS:        R5      UCB address
                       087B  ;                 R4      PCB address
                       087B  ;                 R3      IRP address if UCB is busy
                       087B  ;                 R2      Channel number
                       087B  ;                 R1,R0   Scratch
                       087B  ;
                       087B  ;                 NET$C_IPL
                       087B  ;
                       087B  ;  OUTPUTS:       R3-R0   Garbage
                       087B  ;
                       087B  ;                 All other registers are preserved
                       087B  ;
                       087B  ;-
                       087B  NET$CANCEL:                                             ; Cancel I/O entry point
         50   34 A5    D0  087B          MOVL    UCB$L_VCB(R5),R0                        ; Get VCB address
                 24    13  087F          BEQL    50$                                     ; If EQL then none
                           0881          ;
                           0881          ;
                           0881          ;       Tell the ACP
                           0881          ;
                           0881          ;
             01BC 8F    BB 0881          PUSHR   #^M<R2,R3,R4,R5,R7,R8>                  ; Save regs
                           0885                                                          ;
              57   52   D0 0885          MOVL    R2,R7                                   ; Save channel number
         55    14 A0    D0 0888          MOVL    RCB$L_ACP_UCB(R0),R5                    ; Get the ACP's UCB
              58   36   3C 088C          MOVZWL  #MSG$_PATHLOST,R8                       ; Setup mailbox message code
              52   06   D0 088F          MOVL    #6,R2                                   ; No. of bytes to be entered
                02DC    30 0892          BSBW    NET$SEND_MBX                            ; Setup the message
               09 50    E9 0895          BLBC    R0,30$                                  ; Br on error -- ignore it
         83    60 A4    D0 0898          MOVL    PCB$L_PID(R4),(R3)+                     ; Enter the PID
              83   57   B0 089C          MOVW    R7,(R3)+                                ; Enter channel
                   9E   16 089F          JSB     @(SP)+                                  ; Send the message
                           08A1                                                          ;
             01BC 8F    BA 08A1  30$:    POPR    #^M<R2,R3,R4,R5,R7,R8>                  ; Restore regs
                           08A5  50$:    ;
                           08A5          ;
                           08A5          ;       If the unit is busy then it must be a bug sinc NET$STARTIO never
                           08A5          ;       allows an I/O queue to build on the UCB
                           08A5          ;
                           08A5          ;
      01 64 A5    08    E0 08A5          BBS     #UCB$V_BSY,UCB$W_STS(R5),100$           ; Done if UCB is not busy
                        05 08AA          RSB                                             ; Done
                           08AB                                                          ;
                           08AB  100$:   BUG_CHECK NETNOSTATE,FATAL                      ; Our UCB assumptions are wrong
                           08AF
                        08AF  .SBTTL  NET$PURG_RUN    - Cleanup XWB to exit RUN state
                        08AF  ;+
                        08AF  ;
                        08AF  ;  The receiver and transmitter are run-down on both the DATA and INT/LS
                        08AF  ;  LSB's.
                        08AF  ;
                        08AF  ;  It is assumed that this routine is called as a result of a call from one
                        08AF  ;  of the state transition action routines and that there will be a state
                        08AF  ;  transition out of the RUN state as the event processing is completed.  This
                        08AF  ;  is because certain processing -- such as the setting and clearing of XWB
                        08AF  ;  flags -- is assumed to be done as part of the state transition processing
                        08AF  ;  and is therefore done by this routine.
                        08AF  ;
                        08AF  ;
                        08AF  ;  INPUTS:        R5      XWB address; low bit set if no XWB
                        08AF  ;                 R0      Scratch
                        08AF  ;
                        08AF  ;  OUTPUTS:       R0      Garbage
                        08AF  ;
                        08AF  ;                 All other registers are preserved
                        08AF  ;
                        08AF  ;-
                        08AF  NET$PURG_RUN::                                          ; Leave the RUN state
         01DE 8F    BB  08AF          PUSHR   #^M<R1,R2,R3,R4,R6,R7,R8>               ; Save regs
                        08B3                                                          ;
   05 0E A5    04   E0  08B3          BBS     #XWB$V_STS_CON,XWB$W_STS(R5),20$        ; If BS, not in RUN format
               D8   10  08B8          BSBB    DRAIN_XMT                               ; Drain the transmitter
             0064   30  08BA          BSBW    DRAIN_RCV                               ; Drain the receiver
                        08BD                                                          ;
         01DE 8F    BA  08BD  20$:    POPR    #^M<R1,R2,R3,R4,R6,R7,R8>               ; Restore regs
                    05  08C1          RSB
                        08C2
                        08C2
                        08C2  DRAIN_XMT:                                      ; Drain the xmitter
                        08C2          ;
                        08C2          ;       All transmit CXB's are detached and eventually deallocated.
                        08C2          ;       All transmit IRP's are sent to I/O Post with disconenct status.
                        08C2          ;       The LSB transmitter state variables are updated to reflect an
                        08C2          ;       idle transmitter.
                        08C2          ;
                        08C2          ;
                        08C2          ;       Inputs:     R8,R7   Scratch
                        08C2          ;                   R5      XWB address
                        08C2          ;                   R4-R0   Scratch
                        08C2          ;
                        08C2          ;       Outputs:    R8,R4-R0  garbage.
                        08C2          ;
                        08C2          ;                   All other registers are preserved.
                        08C2          ;
                        08C2          ;
             F9FB   30  08C2          BSBW    NET$MAP_R_REASON                ; Map disconnect reason to status
     50     02 A0   3C  08C5          MOVZWL  REASON_Q_SS(R0),R0              ; Get proper I/O status code
               51   D4  08C9          CLRL    R1                              ; IOSB second longword
     58   00D4 C5   9E  08CB          MOVAB   XWB$T_LI(R5),R8                 ; Get the LS/INT LSB
               11   10  08D0          BSBB    10$                             ; Do it
     58   00A4 C5   9E  08D2          MOVAB   XWB$T_DT(R5),R8                 ; Get the DATA LSB
               0A   10  08D7          BSBB    10$                             ; Do it
```

```
                   08D9                 ;
                   08D9                 ;
                   08D9                 ;       Simulate an ACK on each active CXB thus causing them to be
                   08D9                 ;       deallocated.  This will lead to a false value for LSB$B_X_REQ,
                   08D9                 ;       but this is tolerable since we're about to exit the RUN state.
                   08D9                 ;
                   08D9                 ;          (only the DATA subchannel has CXB's attached to the LSB).
                   08D9                 ;
                   08D9                 ;
54    0D A8    9A  08D9                 MOVZBL  LSB$B_X_CXBACT(R8),R4        ; Number of active Xmt CXB's
         03    13  08DD                 BEQL    5$                           ; If EQL then none
      F71E'    30  08DF                 BSBW    NET$ACK_XMT_SEGS             ; "ACK" each segment release CXB's
               05  08E2       5$:       RSB                                  ; Done
                   08E3
                   08E3
   7E    50    7D  08E3       10$:      MOVQ     R0,-(SP)                    ; Save IOSB image
                   08E6                 ;
                   08E6                 ;
                   08E6                 ;       Update Xmitter state variables
                   08E6                 ;
                   08E6                 ;
D2 A8    68    B0  08E6                 MOVW     LSB$W_LUX(R8),LSB$W_LNX(R8) ; Pretend we've sent all packets
06 A8    68    B0  08EA                 MOVW     LSB$W_LUX(R8),LSB$W_HAR(R8) ; Pretend all packets were ACK'd
08 A8    68    B0  08EE                 MOVW     LSB$W_LUX(R8),LSB$W_HAA(R8) ; No further ACK's expected
04 A8    68    B0  08F2                 MOVW     LSB$W_LUX(R8),LSB$W_HXS(R8) ; No further packets to send
                   08F6                 ;
                   08F6                 ;
                   08F6                 ;       Join the Xmitter's IRP lists, setup each IRP with new I/O status
                   08F6                 ;
                   08F6                 ;
                   08F6                 ASSUME  IRP$L_IOQFL  EQ  0
                   08F6
51    14 A8    9E  08F6                 MOVAB   LSB$L_X_IRP(R8),R1           ; Get spent IRP listhead
         0D    11  08FA                 BRB     40$
      10 A8    D4  08FC       20$:      CLRL    LSB$L_X_PND(R8)              ; Detach pending IRP list
   61    50    D0  08FF                 MOVL    R0,(RT)                      ; Attach it to end of spent IRP list
   51    50    D0  0902       30$:      MOVL    R0,R1                        ; Update last IRP pointer
38 A0    6E    7D  0905                 MOVQ    (SP),IRP$L_IOST1(R0)         ; Overwrite status
   50    61    D0  0909       40$:      MOVL    (R1),R0                      ; Get next IRP
         F4    12  090C                 BNEQ    30$                          ; If NEQ, IRP was found
50    10 A8    D0  090E                 MOVL    LSB$L_X_PND(R8),R0           ; Get pending IRP list
         E8    12  0912                 BNEQ    20$                          ; If NEQ, not empty
53    14 A8    D0  0914                 MOVL    LSB$L_X_IRP(R8),R3           ; Get first IRP
         03    13  0918                 BEQL    100$                         ; If EQL, none
      F6E3'    30  091A                 BSBW    NET$XMT_DONE                 ; Complete all Xmt IRPs
                   091D                                                      ;
   50    8E    7D  091D       100$:     MOVQ    (SP)+,R0                     ; Restore stack and R0
               05  0920                 RSB                                  ; Done
                   0921
                   0921
                   0921       DRAIN_RCV:                                     ; Drain the receiver
                   0921                 ;
                   0921                 ;       All receive CXB's are detached and deallocated.
                   0921                 ;
                   0921                 ;       All receive IRP's are sent to I/O Post with disconnect status.
                   0921                 ;       For each LSB, LSB$B_R_CXBQUO is zeroed to prevent further CXB's
                   0921                 ;       from being received.
```

```
                          0921               ;
                          0921               ;
                          0921               ;       Inputs:     R8      Scratch
                          0921               ;                   R5      XWB address
                          0921               ;                   R3      Scratch
                          0921               ;                   R1-R0   Scratch
                          0921               ;
                          0921               ;       Outputs:    R8,R3,R1,R0  garbage.
                          0921               ;
                          0921               ;                   All other registers are preserved.
                          0921               ;
                          0921               ;
             F99C    30   0921               BSBW    NET$MAP_R_REASON         ; Map disconnect reason to status
     50   02 A0      3C   0924               MOVZWL  REASON_Q_SS(R0),R0       ; Get proper I/O status code
               51    D4   0928               CLRL    R1                       ; IOSB second longword
   58   00D4 C5      9E   092A               MOVAB   XWB$T_LI(R5),R8          ; Get the LS/INT LSB
               05    10   092F               BSBB    10$                      ; Do it
   58   00A4 C5      9E   0931               MOVAB   XWB$T_DT(R5),R8          ; Get the DATA LSB
        7E     50    7D   0936      10$:     MOVQ    R0,-(SP)                 ; Save IOSB image
                          0939               ;
                          0939               ;
                          0939               ;       Drain Receive CXB list
                          0939               ;
                          0939               ;
            29 A8    94   0939               CLRB    LSB$B_R_CXBQUO(R8)       ; Prevent further receives
            28 A8    94   093C               CLRB    LSB$B_R_CXBCNT(R8)       ; Zero the CXB  in use count
     50     20 A8    D0   093F               MOVL    LSB$L_R_CXB(R8),R0       ; Get first CXB in list
               0E    13   0943               BEQL    40$                      ; If EQL then none
            20 A8    D4   0945               CLRL    LSB$L_R_CXB(R8)          ; Detach entire CXB chain from LSB
                          0948                                                ;
            10 A0    DD   0948      30$:     PUSHL   CXB$L_LINK(R0)           ; Save ptr to next CXB
             0400    30   094B               BSBW    NET$DEALLOCATE           ; Deallocate block in R0
               50  8ED0   094E               POPL    R0                       ; Get the next CXB
                          0951                                                ;
               F5    12   0951               BNEQ    30$                      ; If NEQ then loop, else no CXB
                          0953      40$:     ;
                          0953               ;
                          0953               ;       Complete all Rcv IRP's with mapped disconnect status code
                          0953               ;
                          0953               ;
     53     1C A8    D0   0953               MOVL    LSB$L_R_IRP(R8),R3       ; Get next Rcv IRP
               0C    13   0957               BEQL    50$                      ; If EQL then none
            32 A3    B4   0959               CLRW    IRP$W_BCNT(R3)           ; No bytes xferred
   38 A3       6E    7D   095C               MOVQ    (SP),IRP$L_IOST1(R3)     ; Setup I/O status
            F69D'    30   0960               BSBW    NET$RCV_DONE             ; Complete the receive
               EE    11   0963               BRB     40$                      ; Loop
                          0965                                                ;
     50        8E    7D   0965      50$:     MOVQ    (SP)+,R0                 ; Restore regs
               05         0968               RSB                              ; Done
                          0969
```

```
                        0969  .SBTTL  NET$ACP_COMM    - Entry for ACP communication
                        0969  ;++
                        0969  ;
                        0969  ;  This routine is called by the ACP for change of status notification
                        0969  ;  including process exit, logical-link "ownership" changes, and datalink
                        0969  ;  transitions.
                        0969  ;
                        0969  ;
                        0969  ;  CALLING SEQUENCE:
                        0969  ;
                        0969  ;          JSB   @CRB$L_INTD+VEC$L_START  at IPL 0
                        0969  ;
                        0969  ;  INPUTS: R5     NET UCB address.
                        0969  ;          R4-R1 Function specific -- see individual action routine preambles
                        0969  ;          R0    Function code as follows:
                        0969  ;
                        0969  ;                  NETUPD$_CONNECT   - Pass NCB to Declared Name mailbox
                        0969  ;                  NETUPD$_PROCRE    - Process created to received connect
                        0969  ;                  NETUPD$_ABORT     - Process couldn't start
                        0969  ;                  NETUPD$_EXIT      - Started process is exiting
                        0969  ;
                        0969  ;                  NETUPD$_DLL_ON    - Datalink has come online - post a receive
                        0969  ;                  NETUPD$_DLL_DLE   - Datalink online for service fcts
                        0969  ;                  NETUPD$_REACT_RCV - Reactivate Datalink receiver
                        0969  ;                  NETUPD$_SEND_HELLO - Force datalink to send a hello message
                        0969  ;
                        0969  ;                  NETUPD$_CRELNK    - Create a logical-link control structure
                        0969  ;                  NETUPD$_DSCLNK    - Graceful disconnect of single link
                        0969  ;                  NETUPD$_ABOLNK    - Force immediate disconnect of all links
                        0969  ;
                        0969  ;                  NETUPD$_BRDCST    - Broadcast mailbox message
                        0969  ;                  NETUPD$_REPLY     - Reply to associated mailbox
                        0969  ;
                        0969  ;  OUTPUTS:  R0 Status
                        0969  ;
                        0969  ;           All other registers are preserved.
                        0969  ;
                        0969  ;--
              0000000C  0969          R3_OFF = 4*3
              00000010  0969          R4_OFF = 4*4
              00000014  0969          R5_OFF = 4*5
                        0969
                        0969                  .ENABL  LSB
                        0969
                        0969  NET$ACP_COMM::                           ; ACP entry point
                        0969          SETIPL  UCB$B_FIPL(R5)           ; Raise IPL to synch access to structures
                        096D
    07FF 8F   BB        096D          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10>  ; Save regs
                        0971
    5A   5E   D0        0971          MOVL    SP,R10                   ; Save ptr to saved R0
         0B   10        0974          BSBB    20$                      ; Dispatch on fct code
    6E   50   D0        0976          MOVL    R0,(SP)                  ; Overlay return code
                        0979
    07FF 8F   BA        0979          POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10>  ; Restore regs
                        097D
                        097D          SETIPL  #0                       ; Restore IPL
              05        0980          RSB
```

```
                  0981
                  0981  20$:    $DISPATCH  R0,TYPE=B,-           ; Case on function code
                  0981          <-
                  0981              <NETUPD$_CONNECT,    DECLARE>,-   ; Pass NCB to Declared Name mailbox
                  0981              <NETUPD$_PROCRE,     PROCRE>,-    ; Process created to rcv connect
                  0981              <NETUPD$_ABORT,      ABORT>,-     ; Abort single link for given process
                  0981              <NETUPD$_EXIT,       EXIT>,-      ; Started process is exiting
                  0981              -                                 ;
                  0981              <NETUPD$_CRELNK,     CRE_LNK>,-   ; Create a logical-link
                  0981              <NETUPD$_DSCLNK,     DISC_ONE>,-  ; Disconnect single logical-link
                  0981              <NETUPD$_ABOLNK,     ABORT_ALL>,-; Abort all logical-links
                  0981              -                                 ;
                  0981              <NETUPD$_BRDCST,     BRDCST>,-    ; Broadcast mailbox message
                  0981              <NETUPD$_REPLY,      REPLY>,-     ; Send general mailbox message
                  0981              <NETUPD$_DLL_ON,     DLLTRN>,-    ; Datalink made into "on" state
                  0981              -                                 ;
                  0981          >
      018F    31  099B          BRW     UNKNOWN                       ; Let lowest level handle this
                  099E
                  099E  ;+
                  099E  ;  PROCRE   - Process started due to CI received
                  099E  ;
                  099E  ;  INPUTS:  R5  NET UCB address.
                  099E  ;           R4  Scratch
                  099E  ;           R3  Local link number.
                  099E  ;           R2  Scatch
                  099E  ;           R1  PID of process
                  099E  ;
                  099E  ;-
        50    D4  099E  PROCRE: CLRL    R0                            ; Setup for "no PID" match
      009D    30  09A0          BSBW    200$                          ; Get XWB
        11    12  09A3          BNEQ    40$                           ; Done if NEQ
  34 A5 51    D0  09A5          MOVL    R1,XWB$L_PID(R5)              ; Set PID of process allowed
                  09A9                                                ; to complete the connect
        0B    11  09A9          BRB     40$                           ; Done
                  09AB
                  09AB  ;+
                  09AB  ;  ABORT    - Abort single logical-link for a given process
                  09AB  ;
                  09AB  ;  INPUTS:  R5  NET UCB address.
                  09AB  ;           R4  Scratch
                  09AB  ;           R3  Local link number.
                  09AB  ;           R2  Disconnect reason code
                  09AB  ;           R1  PID of process (zero if process not started)
                  09AB  ;
                  09AB  ;-
     50 51    D0  09AB  ABORT:  MOVL    R1,R0                         ; Setup PID (could be zero)
      008F    30  09AE          BSBW    200$                          ; Get XWB
        03    12  09B1          BNEQ    40$                           ; Done if NEQ
      0067    30  09B3          BSBW    180$                          ; Enter DIS state
     50 01    D0  09B6  40$:    MOVL    S^#SS$_NORMAL,R0              ; Report success
          05      09B9          RSB                                   ; Done
                  09BA
                  09BA  ;+
                  09BA  ;  EXIT     - A formerly started process has exited
                  09BA  ;
                  09BA  ;  INPUTS:  R5  NET UCB address
```

```
                         09BA ;                 R4  Scratch
                         09BA ;                 R3  Scratch
                         09BA ;                 R2  Disconnect reason code
                         09BA ;                 R1  PID of process
                         09BA ;
                         09BA ;-
   55    34 A5     D0    09BA EXIT:     MOVL    UCB$L_VCB(R5),R5        ; Get RCB
            1B     13    09BE           BEQL    80$                     ; Br if not mounted
   55    24 A5     D0    09C0           MOVL    RCB$L_PTR_LTB(R5),R5    ; Get LTB
            15     13    09C4           BEQL    80$                     ; Br if its not there
   55    E0 A5     DE    09C6           MOVAL   -XWB$L_LINK -           ;
                         09CA                   +LTB$L_XWB(R5),R5       ; Setup for scan
   55    2C A5     D0    09CA 60$:      MOVL    XWB$L_LINK(R5),R5       ; Get next XWB
            0B     13    09CE           BEQL    80$                     ; If EQL then end of list
      50    51     D0    09D0           MOVL    R1,R0                   ; Copy process PID
            73     10    09D3           BSBB    210$                    ; Check process access to XWB via PID
            F3     12    09D5           BNEQ    60$                     ; If NEQ then something wrong
            44     10    09D7           BSBB    180$                    ; Disconnect the link
            EF     11    09D9           BRB     60$                     ; Continue
      50    01     D0    09DB 80$:      MOVL    S^#SS$_NORMAL,R0        ; Success
                   05    09DE           RSB                             ; Done
                         09DF
                         09DF
                         09DF ;+
                         09DF ;  CRE_LNK   -   Create a single logical-link
                         09DF ;
                         09DF ;  INPUTS:   R5  NET UCB address.
                         09DF ;            R4  Scratch
                         09DF ;            R3  Logical-link's remote node address
                         09DF ;            R2  Scratch
                         09DF ;            R1  PID of process allowed to access link
                         09DF ;
                         09DF ;  OUTPUTS:  R0 XWB address, high bit clear => failure code
                         09DF ;-
                         09DF CRE_LNK:                                  ; Create single logical-link
        01E4       30    09DF           BSBW    NET$CREATE_XWB          ; Create the structure
         07 50     E9    09E2           BLBC    R0,10$                  ; If LBC, failed
   34 A5    51     D0    09E5           MOVL    R1,XWB$L_PID(R5)        ; Setup PID
      50    55     D0    09E9           MOVL    R5,R0                   ; Setup XWB address
                   05    09EC 10$:      RSB                             ; Done
                         09ED
                         09ED ;+
                         09ED ;  DISC_ONE -    Disconnect a single logical-link
                         09ED ;
                         09ED ;  INPUTS:   R5  NET UCB address.
                         09ED ;            R4  Scratch
                         09ED ;            R3  Local link number.
                         09ED ;            R2  Disconnect reason code
                         09ED ;            R1  Logical-link's remote node address
                         09ED ;-
                         09ED DISC_ONE:                                 ; Disconnect single logical-link
        02A7       30    09ED           BSBW    XWB_LOCLNK              ; Find the logical-link XWB
      50    14     3C    09F0           MOVZWL  S^#SS$_BADPARAM,R0      ; Assume no such link exists
         11 55     E8    09F3           BLBS    R5,120$                 ; If LBS then XWB was not found
         3A A5     B5    09F6           TSTW    XWB$W_REMNOD(R5)        ; Remote node 0?
            06     13    09F9           BEQL    100$                    ; If so, ignore node check
   3A A5    51     B1    09FB           CMPW    R1,XWB$W_REMNOD(R5)     ; Same remote node ?
```

```
              06   12  09FF               BNEQ    120$                    ; If not, return error
            0019   30  0A01  100$:        BSBW    180$                    ; Disconnect the link
        50    01   D0  0A04               MOVL    S^#SS$_NORMAL,R0        ; Success
                   05  0A07  120$:        RSB
                       0A08
                       0A08  ;+
                       0A08  ;  ABORT_ALL -  Abort all logical-links
                       0A08  ;
                       0A08  ;  INPUTS:  R5  NET UCB address
                       0A08  ;           R4  Scratch
                       0A08  ;           R3  Scratch
                       0A08  ;           R2  Scratch
                       0A08  ;           R1  Ptr to LTB
                       0A08  ;
                       0A08  ;-
                       0A08  ABORT_ALL:                                   ; Abort all logical-links
   55    E0 A1     DE  0A08               MOVAL   -XWB$L_LINK -           ;
                       0A0C                       +LTB$L_XWB(R1),R5       ; Prepare for scan
   55    2C A5     D0  0A0C  140$:        MOVL    XWB$L_LINK(R5),R5       ; Get next XWB
              07   13  0A10               BEQL    160$                    ; If NEQ then got one
        52    08   B0  0A12               MOVW    #NET$C_DR_THIRD,R2      ; Reason is "third party abort"
              06   10  0A15               BSBB    180$                    ; Mark link to be broken
              F3   11  0A17               BRB     140$                    ; Loop
        50    01   D0  0A19  160$:        MOVL    S^#SS$_NORMAL,R0        ; Success
                   05  0A1C               RSB
                       0A1D
                       0A1D
                       0A1D  ;
                       0A1D  ;  Disconnect the link
                       0A1D  ;
           46 A5   B1  0A1D  180$:        CMPW    XWB$W_X_REASON(R5),-    ; Remote reason been setup yet?
         0064 8F       0A20                       #NET$C_DR_INVALID       ;
              04   12  0A23               BNEQ    190$                    ; If NEQ then yes
     46 A5    52   B0  0A25               MOVW    R2,XWB$W_X_REASON(R5)   ; Enter disconnect reason
           44 A5   B1  0A29  190$:        CMPW    XWB$W_R_REASON(R5),-    ; Local reason been setup yet?
         0064 8F       0A2C                       #NET$C_DR_INVALID       ;
              04   12  0A2F               BNEQ    195$                    ; If NEQ then yes
     44 A5    52   B0  0A31               MOVW    R2,XWB$W_R_REASON(R5)   ;
           F5C8'   30  0A35  195$:        BSBW    NET$MARK_LINK           ; Mark the link to be broken
        50    6A   7D  0A38               MOVQ    (R10),R0                ; Restore R0,R1,R2
   52    08 AA     D0  0A3B               MOVL    8(R10),R2               ;
                   05  0A3F               RSB                             ; Done
                       0A40  ;
                       0A40  ;  Find XWB, verify access rights by PID
                       0A40  ;
            0254   30  0A40  200$:        BSBW    XWB_LOCLNK              ; Find XWB via local link number
              01   95  0A43               TSTB    #1                      ; Clear Z-bit, assuming error
           0A 55   E8  0A45               BLBS    R5,220$                 ; If LBS then no XWB
     34 A5    50   D1  0A48  210$:        CMPL    R0,XWB$L_PID(R5)        ; Is the process the owner ?
              04   12  0A4C               BNEQ    220$                    ; If NEQ then no
              03   91  0A4E               CMPB    #XWB$C_STA_CIR,-        ;
           1E A5       0A50                       XWB$B_STA(R5)           ; Verify state
                   05  0A52  220$:        RSB
                       0A53
                       0A53
                       0A53  ;+
                       0A53  ;  BRDCST   -   Broadcast a mailbox message
```

```
                        0A53  ;
                        0A53  ;  INPUTS:  R5  NET UCB address
                        0A53  ;           R4  Ptr to mailbox msg text
                        0A53  ;           R3  Associated mailbox mask (0 if broadcast to all mailboxes)
                        0A53  ;           R2  Mailbox msg code
                        0A53  ;           R1  Scratch
                        0A53  ;
                        0A53  ;-
                        0A53  BRDCST:                                         ; Broadcast mailbox message
                        0A53              ;
                        0A53              ; & Code to set up R3 here will move to NETACP, eventually
                        0A53              ;
  58   F791 CF     9E   0A53              MOVAB   MBX_TABLE,R8                ; Point to filter mapping table
         53 88     D0   0A58  300$:       MOVL    (R8)+,R3                    ; Get next mask
            05     13   0A5B              BEQL    320$                        ; If EQL at end of table - take the msg
         52 88     B1   0A5D              CMPW    (R8)+,R2                    ; Is this the msg being sent?
            F6     12   0A60              BNEQ    300$                        ; If NEQ no - loop; else, R3 has bit
         58 52     D0   0A62  320$:       MOVL    R2,R8                       ; Transfer msg type code
            00     DD   0A65              PUSHL   #0                          ; Assume no message text
         57 5E     D0   0A67              MOVL    SP,R7                       ; Point to it
            54     D5   0A6A              TSTL    R4                          ; Any message text?
            26     13   0A6C              BEQL    400$                        ; If EQL no, goto end of loop
         52 64     9A   0A6E              MOVZBL  (R4),R2                     ; Get count field value
            52     D6   0A71              INCL    R2                          ; Inc to get total string size
         57 54     D0   0A73              MOVL    R4,R7                       ; Setup stable string pointer
            1C     11   0A76              BRB     400$                        ; Jump to end of loop
                        0A78
            53     D5   0A78  340$:       TSTL    R3                          ; Will everyone take this message?
            06     13   0A7A              BEQL    360$                        ; If EQL yes
   44 A5    53     D3   0A7C              BITL    R3,UCB$L_DEVDEPEND(R5)      ; Can this UCB take this message?
            12     13   0A80              BEQL    400$                        ; If EQL no - don't even try to send
            2C     BB   0A82  360$:       PUSHR   #^M<R2,R3,R5>               ; Save regs
          00EA     30   0A84              BSBW    NET$SEND_MBX                ; Call co-routine to setup the message
         08 50     E9   0A87              BLBC    R0,380$                     ; If LBC then error
         51 57     D0   0A8A              MOVL    R7,R1                       ; Get message pointer
          02DB     30   0A8D              BSBW    NET$MOV_CSTR                ; Move the string with count field
            9E     16   0A90              JSB     @(SP)+                      ; Complete the message
            2C     BA   0A92  380$:       POPR    #^M<R2,R3,R5>               ; Recover regs
  55     30 A5     D0   0A94  400$:       MOVL    UCB$L_LINK(R5),R5           ; Get next UCB
            DE     12   0A98              BNEQ    340$                        ; If NEQ then got one
            8E     D5   0A9A              TSTL    (SP)+                       ; Fix the stack
         50 01     D0   0A9C              MOVL    S^#SS$_NORMAL,R0            ; Exit with success
                   05   0A9F              RSB
                        0AA0
                        0AA0
                        0AA0  ;+
                        0AA0  ;  REPLY     -  Send general message to assocaited mailbox
                        0AA0  ;
                        0AA0  ;  INPUTS:  R5  NET UCB address
                        0AA0  ;           R4  Ptr to mailbox msg text
                        0AA0  ;           R3  & Associated mailbox mask if NETUPD$_BRDCST (0 if broadcast all)
                        0AA0  ;           R2  Mailbox msg code
                        0AA0  ;           R1  Scratch
                        0AA0  ;
                        0AA0  ;-
         58 52     D0   0AA0  REPLY:      MOVL    R2,R8                       ; Get mailbox message code
            13     11   0AA3              BRB     500$                        ; Continue in common
```

```
                        0AA5
                        0AA5    ;+
                        0AA5    ;  DECLARE  -   Pass NCB to Declared-name mailbox
                        0AA5    ;
                        0AA5    ;  INPUTS:  R5  NET UCB address
                        0AA5    ;           R4  Ptr to NCB counted string
                        0AA5    ;           R3  Scratch
                        0AA5    ;           R2  Scratch
                        0AA5    ;           R1  Scracth
                        0AA5    ;
                        0AA5    ;-
                        0AA5    DECLARE:                                        ; Pass NCB to declare-object mailbox
          01EF    30    0AA5            BSBW    XWB_LOCLNK                      ; Find link's XWB
         29 55    E8    0AA8            BLBS    R5,560$                         ; Br if no XWB
            03    91    0AAB            CMPB    #XWB$C_STA_CIR,-                ; Must be in CIR state
         1E A5          0AAD                    XWB$B_STA(R5)                   ;
            23    12    0AAF            BNEQ    560$                            ; If not then cannot redirect connect
   34 A5    51    D0    0AB1            MOVL    R1,XWB$L_PID(R5)                ; Set PID of process
      58    32    3C    0AB5            MOVZWL  #MSG$_CONNECT,R8                ; Setup mailbox message type
   54    10 AA    7D    0AB8    500$:   MOVQ    R4_OFF(R10),R4                  ; Get mbx message and UCB addresses
      52    54    D0    0ABC            MOVL    R4,R2                           ; Copy msg pointer
            03    13    0ABF            BEQL    520$                            ; If EQL then no text
      52    62    9A    0AC1            MOVZBL  (R2),R2                         ; Set count of bytes to be sent
          00AA    30    0AC4    520$:   BSBW    NET$SEND_MBX                    ; Prepare to send mailbox message
         0D 50    E9    0AC7            BLBC    R0,580$                         ; Br on error
      51    54    D0    0ACA            MOVL    R4,R1                           ; Copy NCB pointer
            03    13    0ACD            BEQL    540$                            ; Skip if null
          0299    30    0ACF            BSBW    NET$MOV_CSTR                    ; Move counted string into buffer
            9E    16    0AD2    540$:   JSB     @(SP)+                          ; Complete writing mailbox
      50    01    D0    0AD4    560$:   MOVL    S^#SS$_NORMAL,R0                ; Success
                  05    0AD7    580$:   RSB                                     ; Done
                        0AD8
                        0AD8
                        0AD8    ;+
                        0AD8    ;  DLLTRN   -   Datalink state transition
                        0AD8    ;
                        0AD8    ;  INPUTS:  R5  NET UCB address
                        0AD8    ;           R4  Scratch
                        0AD8    ;           R3  Scratch
                        0AD8    ;           R2  Scratch
                        0AD8    ;           R1  Ptr to datalink's LPD
                        0AD8    ;
                        0AD8    ;-
   52    34 A5    D0    0AD8    DLLTRN: MOVL    UCB$L_VCB(R5),R2                ; Get RCB
            01    91    0ADC            CMPB    #LPD$C_LOC_INX,-                ; Is this the local LPD
         20 A1          0ADE                     LPD$B_PTH_INX(R1)              ;
            4B    12    0AE0            BNEQ     UNKNOWN                        ; If not, branch
                        0AE2
            3F    BB    0AE2            PUSHR   #^M<R0,R1,R2,R3,R4,R5>          ; Save regs
      53    55    D0    0AE4            MOVL    R5,R3                           ; Copy UCB address
   55    30 A2    D0    0AE7            MOVL    RCB$L_PTR_TQE(R2),R5            ; Get TQE
            02    E0    0AEB            BBS     #TQE$V_REPEAT,-                 ; Br if timer is in use
      3B 0B A5          0AED                    TQE$B_RQTYPE(R5),600$           ;
            05    90    0AF0            MOVB    #TQE$C_SSREPT,-                 ; Set for system subroutine repeat
         0B A5          0AF2                    TQE$B_RQTYPE(R5)                ;
       0000'CF    9E    0AF4            MOVAB   W^NET$TIMER,-                   ; Set timer handler address
         0C A5          0AF8                    TQE$L_FPC(R5)                   ;
```

```
              14 A5    53    D0  0AFA            MOVL    R3,TQE$L_FR4(R5)          ; Save UCB address
00000000 00989680 8F         7D  0AFE            MOVQ    #10*1000*1000,-           ; 1 tick = 1 sec
                 20 A5           0B08                    TQE$Q_DELTA(R5)           ;
       50    F60A CF         9E  0B0A            MOVAB   W^NET$GL_OFF_DPTFLG,R0    ; Get address of offset to DPT$B_FLAGS
               50    60      C0  0B0F            ADDL    (R0),R0                   ; Make it an address
               60    04      88  0B12            BISB    #DPT$M_NOUNLOAD,(R0)      ; Prevent reload of driver
   50    00000000'GF         7D  0B15            MOVQ    G^EXE$GQ_SYSTIME,R0       ; Set time of first tick
                                 0B1C            DSBINT  #IPL$_TIMER               ; Lower IPL to that of timer service
         00000000'GF         16  0B22            JSB     G^EXE$INSTIMQ             ; Insert into queue
                                 0B28            ENBINT                            ; Restore IPL
                    3F       BA  0B2B  600$:     POPR    #^M<R0,R1,R2,R3,R4,R5>    ;
                                 0B2D                                              ;
                                 0B2D  UNKNOWN:                                    ;
                 F4D0'       30  0B2D            BSBW    TR$UPDATE                 ; Fct code is in R0
                             05  0B30            RSB                               ; Exit with status in R0
                                 0B31
                                 0B31                    .DSABL  LSB
                                 0B31
```

```
                         0B31  .SBTTL  NET$SEND_CS_MBX - Send counted string to mailbox
                         0B31  ;+
                         0B31  ;
                         0B31  ;  A mailbox message is built and sent to the mailbox associated with the UCB
                         0B31  ;  associated with the XWB.  The counted string pointed to by R1 is appended to
                         0B31  ;  the end of the mailbox message.  R2 contains the assumed total count of the
                         0B31  ;  string and may be zero.  If there is no mailbox then the routine is assumed
                         0B31  ;  implicitly successful.
                         0B31  ;
                         0B31  ;
                         0B31  ;  INPUTS:  R8  Mailbox message type code
                         0B31  ;           R5  XWB address
                         0B31  ;           R2  Assumed total length of string (low byte only)
                         0B31  ;           R1  Address of count field of string
                         0B31  ;
                         0B31  ;  OUTPUTS: R2  Zero
                         0B31  ;           R1  Garbage
                         0B31  ;           R0  SS$_NORMAL         if mailbox successfully written
                         0B31  ;               SS$_NOMBX!1        if no associated mailbox or no UCB
                         0B31  ;               Zero               if (R1)+1 NEQ R2  or  R2 GTRU 17
                         0B31  ;               Also see           NET$SEND_MBX for R0 error codes
                         0B31  ;
                         0B31  ;
                         0B31  ;         All other registers are preserved
                         0B31  ;
                         0B31  ;-
                         0B31  NET$SEND_CS_MBX::
               3E    BB  0B31          PUSHR   #^M<R1,R2,R3,R4,R5>      ; Save regs
                         0B33                                           ;
               52    D5  0B33          TSTL    R2                       ; Any bytes in string ?
               06    12  0B35          BNEQ    10$                      ; If NEQ yes, else can't trust R1
   6E     70'AF      9E  0B37          MOVAB   B^50$,(SP)               ; Setup null string ptr
               0F    11  0B3B          BRB     20$                      ; Continue
               50    D4  0B3D  10$:    CLRL    R0                       ; Assume string error
         11    52    D1  0B3F          CMPL    R2,#17                   ; Is count within range
               27    1A  0B42          BGTRU   40$                      ; If not, branch
   53    52    61    83  0B44          SUBB3   (R1),R2,R3               ; Check count field consistency
               53    97  0B48          DECB    R3                       ; Account for count field itself
               1F    12  0B4A          BNEQ    40$                      ; Inconsistent if NEQ
   50   0275 8F      3C  0B4C  20$:    MOVZWL  #SS$_NOMBX!1,R0          ; Assume no UCB or mailbox
     55    10 A5     D0  0B51          MOVL    XWB$L_ORGUCB(R5),R5      ; Get UCB
               14    13  0B55          BEQL    40$                      ; If none, done
            60 A5    D5  0B57          TSTL    UCB$L_AMB(R5)            ; Is there a mailbox ?
               0F    13  0B5A          BEQL    40$                      ; If not, branch
             0012    30  0B5C          BSBW    NET$SEND_MBX             ; Build header (co-routine)
            09 50    E9  0B5F          BLBC    R0,40$                   ; Br on error
     51    04 AE     D0  0B62          MOVL    4(SP),R1                 ; Get string address (note stack)
             0202    30  0B66          BSBW    NET$MOV_CSTR             ; Move string with count field
               9E    16  0B69          JSB     @(SP)+                   ; Close and send mbx message
                         0B6B                                           ;
               3E    BA  0B6B  40$:    POPR    #^M<R1,R2,R3,R4,R5>      ; Recover regs
               52    D4  0B6D          CLRL    R2                       ; String has been consumed
                     05  0B6F          RSB
                         0B70
                     00  0B70  50$:    .BYTE   0                        ; Phony counted string for mailbox
```

```
                         0B71  .SBTTL  NET$SEND_MBX    - Co-routine to send mailbox message
                         0B71  ;+
                         0B71  ;
                         0B71  ;  The first time the routine is entered the associated mailbox is found, a
                         0B71  ;  buffer is allocated for the message, and the mailbox header is built.  When
                         0B71  ;  the routine is re-entered, after a call-back to the co-routine, the message
                         0B71  ;  is closed and sent to the mailbox.
                         0B71  ;
                         0B71  ;  The original entry parameters are given below, the re-entry parameters are
                         0B71  ;  given within the body of the code.
                         0B71  ;
                         0B71  ;
                         0B71  ;  INPUTS:  R8  Mailbox message type code
                         0B71  ;           R5  UCB address
                         0B71  ;           R3  Scratch
                         0B71  ;           R2  Count of bytes co-routine will enter into message
                         0B71  ;           R1  Scratch
                         0B71  ;           R0  Scratch
                         0B71  ;
                         0B71  ;  OUTPUTS: R3  Pointer to next byte in mailbox message to be filled
                         0B71  ;           R2  Address of allocated buffer if R0=SS$_NORMAL
                         0B71  ;           R1  Garbage
                         0B71  ;           R0  SS$_NORMAL  if successful
                         0B71  ;               SS$_NOMBX   if there's no associated mailbox
                         0B71  ;
                         0B71  ;               see NET$ALONONPAGED for additional error status
                         0B71  ;
                         0B71  ;           All other registers are preserved
                         0B71  ;
                         0B71  ;-
                         0B71  NET$SEND_MBX::
                         0B71             ;
                         0B71             ;
                         0B71             ;     Add 24 to the number of bytes the user will enter.  This will
                         0B71             ;     ensure that the allocated block is large enough for COM$DRVDEALMEM
                         0B71             ;     to deallocate -- also creates space for:
                         0B71             ;
                         0B71             ;        12 bytes for  standard buffer header
                         0B71             ;         2 bytes for mailbox msg type code
                         0B71             ;         2 bytes for mailbox  unit number
                         0B71             ;         1 byte  for count field for device name
                         0B71             ;
                         0B71             ;
          52    18    C0 0B71             ADDL    #24,R2                  ; Increase buffer size
     50   28 A5       D0 0B74             MOVL    UCB$L_DDB(R5),R0        ; DDB pointer
     51   14 A0       9E 0B78             MOVAB   DDB$T_NAME(R0),R1       ; Get device name string ptr
                         0B7C                                             ;
                51    DD 0B7C             PUSHL   R1                      ; Save device name string ptr
          51    61    9A 0B7E             MOVZBL  (R1),R1                 ; Get string size
          51    52    C0 0B81             ADDL    R2,R1                   ; Add in remaining bytes
             01AD     30 0B84             BSBW    NET$ALONONPAGED         ; Get the buffer
                51  8ED0 0B87             POPL    R1                      ; Restore device name string ptr
                         0B8A                                             ;
          01    50    E8 0B8A             BLBS    R0,10$                  ; If LBS then okay
                      05 0B8D             RSB                             ; Return with error status in R0
                         0B8E
     53   52    0C    C1 0B8E  10$:       ADDL3   #12,R2,R3               ; Get pointer to start of msg
```

```
        83   58   B0  0B92            MOVW    R8,(R3)+                ; Enter message type code
    83    54 A5   B0  0B95            MOVW    UCB$W_UNIT(R5),(R3)+    ; Enter unit I.D.
           01CF   30  0B99            BSBW    NET$MOV_CSTR            ; Move in device name with count field
       50    01   D0  0B9C            MOVL    S^#SS$_NORMAL,R0        ; Indicate success
             9E   16  0B9F            JSB     @(SP)+                  ; Call co-routine for more bytes
                      0BA1                                            ; Note that R4 is unmodified
                      0BA1            ;
                      0BA1            ;
                      0BA1            ;     On coroutine return:      R5 = UCB address
                      0BA1            ;                               R3 = address of 1st byte past mbx msg
                      0BA1            ;                               R2 = buffer address
                      0BA1            ;
                      0BA1            ;     On return to caller:      R0 = EXE$WRITEMBX status
                      0BA1            ;                               R1-R5 are garbage
                      0BA1            ;
                      0BA1            ;
54    52     0C   C1  0BA1            ADDL3   #12,R2,R4               ; Get start of mbx message
      53     54   C2  0BA5            SUBL    R4,R3                   ; Get length of mbx message
50    0274   8F   3C  0BA8            MOVZWL  #SS$_NOMBX,R0           ; Assume no mailbox
  55      60 A5   D0  0BAD            MOVL    UCB$L_AMB(R5),R5        ; Get mailbox
             06   13  0BB1            BEQL    20$                     ; If EQL then no mailbox
   00000000'GF    16  0BB3            JSB     G^EXE$WRTMAILBOX        ; Send message to mailbox
                      0BB9                                            ;
             50   DD  0BB9       20$: PUSHL   R0                      ; Save return status
50    54     0C   C3  0BBB            SUBL3   #12,R4,R0               ; Get buffer address
           018C   30  0BBF            BSBW    NET$DEALLOCATE          ; Deallocate block in R0
             50 8ED0  0BC2            POPL    R0                      ; Restore reg
                      0BC5                                            ;
                  05  0BC5            RSB                             ; Done
                      0BC6
```

```
                    0BC6  .SBTTL  NET$CREATE_XWB  - Create XWB for logical-link
                    0BC6  ;++
                    0BC6  ;
                    0BC6  ;  An XWB (the logical-link control structure that will eventually be attached
                    0BC6  ;  to an I/O channel CCB$L_WIND field) is allocated and initialized, provided
                    0BC6  ;  that the current maximum logical-link count is not exceeded.  The current
                    0BC6  ;  logical-link count in incremented.
                    0BC6  ;
                    0BC6  ;  No local link address is assigned, and the XWB is not linked into the LTB.
                    0BC6  ;
                    0BC6  ;
                    0BC6  ;  INPUTS:       R5      NET UCB Address
                    0BC6  ;                R3      Remote node address
                    0BC6  ;                R0      Scratch
                    0BC6  ;
                    0BC6  ;  OUTPUTS:      R5      Address of XWB if successful, otherwise LBS
                    0BC6  ;                R0      Status
                    0BC6  ;
                    0BC6  ;                All other registers are preserved
                    0BC6  ;
                    0BC6  ;--
                    0BC6  NET$CREATE_XWB::                                        ; Get idle XWB
          1E    BB  0BC6          PUSHR   #^M<R1,R2,R3,R4>                        ; Save regs to be used
                    0BC8          ;
                    0BC8          ;
                    0BC8          ;     Make sure we are not over our limit (MCOUNT = current links + 1).
                    0BC8          ;
                    0BC8          ;
50   027C 8F    3C  0BC8          MOVZWL #SS$_NOLINKS,R0                          ; Assume failure
  52   34 A5    D0  0BCD          MOVL    UCB$L_VCB(R5),R2                        ; Point to RCB
          30    13  0BD1          BEQL    13$                                     ; If EQL then no RCB
  51   54 A2    3C  0BD3          MOVZWL  RCB$W_MCOUNT(R2),R1                     ; Get current Mount Count
          2A    13  0BD7          BEQL    13$                                     ; If EQL, NETACP shutting down
 58 A2    51    B1  0BD9          CMPW    R1,RCB$W_MAX_LNK(R2)                    ; Is new link allowed ?
          19    1A  0BDD          BGTRU   12$                                     ; If not, branch
51   017C 8F    3C  0BDF          MOVZWL #XWB_C_LEN,R1                            ; Get size of XWB
       013C     30  0BE4          BSBW    NET$ALONPGD_Z                           ; Allocate the block and zero it
                    0BE7                                                          ;  to initialize most fields
      51  52    D0  0BE7          MOVL    R2,R1                                   ; Save XWB pointer
  52   34 A5    D0  0BEA          MOVL    UCB$L_VCB(R5),R2                        ; Point to RCB
     55   51    D0  0BEE          MOVL    R1,R5                                   ; Use standard XWB pointer
  54   08 AE    B0  0BF1          MOVW    8(SP),R4                                ; Get dst node address
       10 50    E8  0BF5          BLBS    R0,15$                                  ; Br if successful
                    0BF8  12$:    BUMP    W,RCB$W_CNT_XRE(R2)                     ; Account for resource error
     55   01    D0  0C03  13$:    MOVL    #1,R5                                   ; Invalidate XWB ptr
          14    11  0C06          BRB     100$                                    ; Done
                    0C08  15$:    ;
                    0C08          ;
                    0C08          ;     Initialize the XWB and bump RCB mount count.
                    0C08          ;
                    0C08          ;
          15    10  0C08          BSBB    INIT_XWB                                ; Init XWB
009E C2   54 A2 B1  0C0A          CMPW    RCB$W_MCOUNT(R2),RCB$W_CNT_MLL(R2) ; New max active links value?
          04    1B  0C10          BLEQU   30$                                     ; If LEQU then no
     009E C2    B6  0C12          INCW    RCB$W_CNT_MLL(R2)                       ; Bump max active link count
                    0C16                                                          ; (#links = MCOUNT-1)
       54 A2    B6  0C16  30$:    INCW    RCB$W_MCOUNT(R2)                        ; Account for new link
```

```
           50   01   D0  0C19            MOVL    #1,R0                                ; Success
                1E   BA  0C1C  100$:     POPR    #^M<R1,R2,R3,R4>                     ; Restore regs
                     05  0C1E            RSB                                          ; Done
                         0C1F
                         0C1F  INIT_XWB:                                                        ; Initialize XWB.
         1F A5   08  90  0C1F            MOVB    #NET$C_IPL,        XWB$B_FIPL(R5)            ; Setup fork IPL
         0A A5   1C  90  0C23            MOVB    #DYN$C_NDB,        XWB$B_TYPE(R5)            ; Setup structure type
         1E A5   00  90  0C27            MOVB    #XWB$C_STA_CLO,    XWB$B_STA(R5)             ; Init logical-link state
         0E A5   10  B0  0C2B            MOVW    #XWB$M_STS_CON,    XWB$W_STS(R5)             ; Init the status word
      1C A5   0200 8F  B0  0C2F          MOVW    #XWB$M_FLG_CLO,    XWB$W_FLG(R5)             ; Init FLG bits
      44 A5   0064 8F  B0  0C35          MOVW    #NET$C_DR_INVALID,XWB$W_R_REASON(R5)         ; Init rcv'd discon reason
      46 A5   0064 8F  B0  0C3B          MOVW    #NET$C_DR_INVALID,XWB$W_X_REASON(R5)         ; Init xmt'd discon reason
         3A A5   54  B0  0C41            MOVW     R4,               XWB$W_REMNOD(R5)          ; Setup remote node i.d.
         30 A5   52  D0  0C45            MOVL     R2,               XWB$L_VCB(R5)             ; Setup VCB address
   016B C5     80 8F  90  0C49           MOVB    #^X<80>,           XWB$$+ACB$B_RMOD(R5);     Setup Special Kernal AST
0178 C5    00000000'EF  9E  0C4F         MOVAB    NET$KAST,         XWB$$+ACB$L_KAST(R5);      mode and address
                         0C58                                                                   ;
       50    011B C5  9E  0C58           MOVAB    XWB$Q_FREE_CXB(R5),R0                         ; Get free queue address
             60   50  D0  0C5D           MOVL     R0,(R0)                                       ; Init queue header
             60   80  DE  0C60           MOVAL    (R0)+,(R0)                                    ;
                         0C63                                                                   ;
      54 A5    63 A2  9B  0C63           MOVZBW   RCB$B_ECL_RFA(R2),  XWB$W_RETRAN(R5)          ; Set default rexmit's
      56 A5    64 A2  9B  0C68           MOVZBW   RCB$B_ECL_DFA(R2),  XWB$W_DLY_FACT(R5);       Set default delay factor
      58 A5    65 A2  9B  0C6D           MOVZBW   RCB$B_ECL_DWE(R2),  XWB$W_DLY_WGHT(R5);       Set default delay weight
      42 A5    7C A2  B0  0C72           MOVW     RCB$W_ECLSEGSIZ(R2),XWB$W_REMSIZ(R5)          ; Set temp 'seg' size
50 A5   76 A2     01  A1  0C77           ADDW3  #1,RCB$W_TIM_CNI(R2),XWB$W_TIMER(R5)            ; Set inbound connect timer
                         0C7D                                                                   ; (#1 is for clock skew)
                         0C7D            ;
                         0C7D            ;
                         0C7D            ;      Build the route header.
                         0C7D            ;
                         0C7D            ;
       51    015E C5  9E  0C7D           MOVAB    XWB$T_TR3HDR+6(R5),R1       ; Setup route-header pointer
                  71  94  0C82           CLRB     -(R1)                       ; Zero the "visits" field
         71    0E A2  B0  0C84           MOVW     RCB$W_ADDR(R2),-(R1)        ; Enter src node address
             71   54  B0  0C88           MOVW     R4,-(R1)                    ; Enter dst node address
             71   02  90  0C8B           MOVB    #TR3$C_MSG_DATA,-(R1)        ; Enter message type
           0120 C5   51  D0  0C8E        MOVL     R1,XWB$L_PTR_RTHD(R5)       ; Setup route-header pointer
             71   06  D0  0C93           MOVL    #6,-(R1)                     ; Store the route-header size
                  05  0C96               RSB                                  ; Done
                         0C97
```

```
                                 0C97  .SBTTL  XWB_LOCLNK      - Get XWB via local link number
                                 0C97  ;+
                                 0C97  ;
                                 0C97  ;  INPUTS:      R5      Any NET UCB address
                                 0C97  ;               R3      Local link number
                                 0C97  ;
                                 0C97  ;  OUTPUTS:     R5      Address of associated XWB, or low bit set if none
                                 0C97  ;
                                 0C97  ;               All other registers are preserved.
                                 0C97  ;-
                                 0C97  XWB_LOCLNK:                                 ; Get XWB context
                       14   BB   0C97          PUSHR   #^M<R2,R4>                  ; Save reg
                                 0C99                                              ;
             52    34 A5   D0    0C99          MOVL    UCB$L_VCB(R5),R2            ; Get RCB address
                       05   12   0C9D          BNEQ    5$                          ; If NEQ the RCB exists
                  55   01   D0   0C9F          MOVL    #1,R5                       ; Invalidate XWB address
                       02   11   0CA2          BRB     10$                         ; Done
                       03   10   0CA4  5$:     BSBB    NET$XWB_LOCLNK              ; If NEQ Locate the link
                                 0CA6                                              ;
                       14   BA   0CA6  10$:    POPR    #^M<R2,R4>                  ; Restore reg
                            05   0CA8          RSB
                                 0CA9
                                 0CA9
                                 0CA9  .SBTTL  NET$XWB_LOCLNK  - Get XWB via local link number
                                 0CA9  ;++
                                 0CA9  ;
                                 0CA9  ;  The Link Table is located and the slot associated with the specified link
                                 0CA9  ;  number is found.  If this slot contains an XWB then the link sequence number
                                 0CA9  ;  is checked.  If there is a sequence number mismatch, or if there is no
                                 0CA9  ;  active XWB, then the low bit of R5 is set.  Else, the XWB address is stored
                                 0CA9  ;  in R5.
                                 0CA9  ;
                                 0CA9  ;
                                 0CA9  ;  INPUTS:      R5,R4   Scratch
                                 0CA9  ;               R3      Local link number - high order word is clear
                                 0CA9  ;               R2      RCB address
                                 0CA9  ;
                                 0CA9  ;  OUTPUTS:     R5      Address of associated XWB, or low bit set if none
                                 0CA9  ;               R4      LTB (link table) address
                                 0CA9  ;
                                 0CA9  ;               All other registers are preserved.
                                 0CA9  ;
                                 0CA9  ;--
                                 0CA9  NET$XWB_LOCLNK::                            ; Locate XWB via local link number
             54    24 A2   D0    0CA9          MOVL    RCB$L_PTR_LTB(R2),R4        ; Get Link Table pointer
                       1E   13   0CAD          BEQL    20$                         ; Return error if not there
   55    53   FFFFFC00 8F   CB   0CAF          BICL3   #^C<NET$C_MAXLNK>,R3,R5     ; Get link 'index'
                       14   13   0CB7          BEQL    20$                         ; Index '0' isn't used
             04 A4     55   B1   0CB9          CMPW    R5,LTB$W_SLT_TOT(R4)        ; Index within range ?
                       0E   1A   0CBD          BGTRU   20$                         ; If not, branch
          55    10 A445     D0   0CBF          MOVL    LTB$L_SLOTS(R4)[R5],R5      ; Get XWB address
                    09 55   E8   0CC4          BLBS    R5,30$                      ; If LBS then none
             3E A5     53   B1   0CC7          CMPW    R3,XWB$W_LOCLNK(R5)         ; Sequence number match ?
                       03   13   0CCB          BEQL    30$                         ; If so, branch
                  55   01   88   0CCD  20$:    BISB    #1,R5                       ; Flag no associated XWB
                            05   0CD0  30$:    RSB
                            0CD1  .SBTTL  NET$RET_SLOT    - Return logical-link XWB slot if done
                            0CD1  .SBTTL  NET$QUE_XWB     - Queue XWB to NETACP's AQB
                            0CD1  ;++
                            0CD1  ;
                            0CD1  ;  If the XWB is busy then the queue attempt is aborted.  If the XWB is
                            0CD1  ;  not busy then the XWB$V_STS_SOL bit is set to prevent any further XWB use.
                            0CD1  ;
                            0CD1  ;
                            0CD1  ;
                            0CD1  ;  INPUTS:        R5      XWB pointer
                            0CD1  ;
                            0CD1  ;  OUTPUTS:       R0,R1   Zero
                            0CD1  ;
                            0CD1  ;                 All other registers are preserved.
                            0CD1  ;
                            0CD1  ;
                            0CD1  ;--
                            0CD1  NET$RET_SLOT::                                          ; Return logical-link if done
          1E A5    00   91  0CD1                  CMPB    #XWB$C_STA_CLO,XWB$B_STA(R5)    ; In 'closed' state?
                   06   13  0CD5                  BEQL    10$                             ; If so, continue
          1E A5    06   91  0CD7                  CMPB    #XWB$C_STA_DIR,XWB$B_STA(R5)    ; If DIR state then we've sent
                            0CDB                                                          ; the DC msg already
                   11   12  0CDB                  BNEQ    40$                             ; If not, XWB is still active
             0C A5      B5  0CDD  10$:            TSTW    XWB$W_REFCNT(R5)                ; Any references ?
                   0C   12  0CE0                  BNEQ    40$                             ; If NEQ must wait
                            0CE2  ;&              BBS     #XWB$V_FLG_LOCK,XWB$W_FLG(R5),40$; Exit if XWB is locked
     0E A5    0C04 8F   B3  0CE2                  BITW    #XWB$M_STS_ASTPND!-             ; AST pending
                            0CE8                          XWB$M_STS_ASTREQ!-              ; AST requested
                            0CE8                          XWB$M_STS_SOL,XWB$W_STS(R5)     ; Fork block in use
                   04   12  0CE8                  BNEQ    40$                             ; If NEQ, XWB is busy
                   2A   10  0CEA                  BSBB    NET$DRAIN_FREE_CXB              ; Drain CXB free queue
                   03   10  0CEC                  BSBB    NET$QUE_XWB                     ; Queue XWB to NETACP's AQB
                   50   7C  0CEE  40$:            CLRQ    R0                              ; Say ''nothing to xmit''
                        05  0CF0                  RSB                                     ; Done
                            0CF1
                            0CF1
                            0CF1  NET$QUE_XWB::                                           ; Queue XWB to NETACP's AQB
                            0CF1                  ASSUME  IPL$_SYNCH  EQ  NET$C_IPL
                            0CF1
       1F 0E A5    02   E2  0CF1                  BBSS    #XWB$V_STS_SOL,XWB$W_STS(R5),50$; If BS, then queue block in use
                   3C   BB  0CF6                  PUSHR   #^M<R2,R3,R4,R5>                ; Save regs
                            0CF8                                                          ;
         52    30 A5    D0  0CF8                  MOVL    XWB$L_VCB(R5),R2                ; Get RCB
               0C A2    B6  0CFC                  INCW    RCB$W_TRANS(R2)                 ; Account for ACP transaction
         54    10 A2    D0  0CFF                  MOVL    RCB$L_AQB(R2),R4                ; Get AQB
         04 B4     65   0E  0D03                  INSQUE  (R5),@AQB$L_ACPQBL(R4)          ; Queue XWB to AQB
                   0A   12  0D07                  BNEQ    30$                             ; If NEQ then not first
         51    0C A4    D0  0D09                  MOVL    AQB$L_ACPPID(R4),R1             ; Get ACP's PID
       00000000'GF      16  0D0D                  JSB     G^SCH$WAKE                      ; Wake the ACP
                            0D13
                   3C   BA  0D13  30$:            POPR    #^M<R2,R3,R4,R5>                ; Restore regs
                        05  0D15  50$:            RSB                                     ; done
                            0D16
                            0D16
                            0D16
                            0D16  .SBTTL  NET$DRAIN_FREE_CXB      - Drain CXB free queue
                            0D16
                        0D16  NET$DRAIN_FREE_CXB::                                   ; Drain CXB free queue
                        0D16            ;
                        0D16            ;
                        0D16            ;       All registers except for R0 must be preserved.
                        0D16            ;
                        0D16            ;
     50    0118 D5  0F  0D16  10$:      REMQUE  @XWB$Q_FREE_CXB(R5),R0               ; Get next CXB
                 05  1D  0D1B            BVS     20$                                  ; If VS, none left
               002E  30  0D1D            BSBW    NET$DEALLOCATE                       ; Deallocate block in R0
                 F4  11  0D20            BRB     10$                                  ; Loop
                     05  0D22  20$:      RSB                                          ; Done
                         0D23
                         0D23
```

```
                              0D23  .SBTTL  NET$ALONPGD_Z    - Allocate and zero from system pool
                              0D23  .SBTTL  NET$ALONONPAGED - Allocate from system pool
                              0D23  ;++
                              0D23  ;
                              0D23  ;  A buffer is allocated from non-paged pool and its size field is set to
                              0D23  ;  the size requested.  Its type field is set to DYN$C_CXB.
                              0D23  ;
                              0D23  ;
                              0D23  ;  INPUTS:  R2 = Scratch
                              0D23  ;           R1 = Size, in bytes, of block to be allocated
                              0D23  ;           R0 = Scratch
                              0D23  ;
                              0D23  ;  OUTPUTS: R2 = Address of block if successful
                              0D23  ;                Zero if unsuccessful
                              0D23  ;           R0 = Standard VMS status code
                              0D23  ;
                              0D23  ;           All other registers are preserved.
                              0D23  ;
                              0D23  ;--
                              0D23                  .ENABL  LSB
                              0D23  NET$ALONPGD_Z::                                ; Allocate and zero non-paged buffer
                      0F   10 0D23          BSBB    NET$ALONONPAGED                ; Allocate the buffer
                   25 50   E9 0D25          BLBC    R0,20$                         ; If LBC then error
                              0D28                                                 ;
                      3F   BB 0D28          PUSHR   #^M<R0,R1,R2,R3,R4,R5>         ; Save regs
      62   51   00   6E 00 2C 0D2A          MOVC5   #0,(SP),#0,R1,(R2)             ; Zero the entire buffer
                      3F   BA 0D30          POPR    #^M<R0,R1,R2,R3,R4,R5>         ; Restore regs
                              0D32                                                 ;
                      11   11 0D32          BRB     10$                            ; Setup the type and size fields (again)
                              0D34
                              0D34
                              0D34  NET$ALONONPAGED::                              ; Allocate non-paged memory
                              0D34                                                 ;
                      0A   BB 0D34          PUSHR   #^M<R1,R3>                     ; Save regs
               00000000'GF 16 0D36          JSB     G^EXE$ALONONPAGED              ; Allocate memory
                      0A   BA 0D3C          POPR    #^M<R1,R3>                     ; Restore regs
                              0D3E                                                 ;
                   04 50   E8 0D3E          BLBS    R0,10$                         ; If LBS then success
                      52   D4 0D41          CLRL    R2                             ; Zero the buffer pointer
                      08   11 0D43          BRB     20$                            ; Take common exit
                08 A2 51   B0 0D45  10$:    MOVW    R1,CXB$W_SIZE(R2)              ; Set size for deallocation
                      1B   90 0D49          MOVB    #DYN$C_CXB,-                   ;
                   0A A2      0D4B                  CXB$B_TYPE(R2)                 ; Set tentative buffer type
                           05 0D4D  20$:    RSB                                    ; Return with status in R0
                              0D4E
                              0D4E                  .DSABL  LSB
                              0D4E
```

```
                        0D4E  .SBTTL  NET$DEALLOCATE  - Deallocate non-paged pool
                        0D4E  ;+
                        0D4E  ;
                        0D4E  ;  IPL must be NET$C_IPL or lower.
                        0D4E  ;
                        0D4E  ;
                        0D4E  ;  INPUTS:      R0      Address of block
                        0D4E  ;
                        0D4E  ;  OUTPUTS:     R0      Zero
                        0D4E  ;
                        0D4E  ;               All other registers are preserved.
                        0D4E  ;
                        0D4E  ;-
                        0D4E          ASSUME  NET$C_IPL  LE  IPL$_SYNCH       ; Can't deallocate above SYNCH
                        0D4E
                        0D4E  NET$DEALLOCATE::                          ; Deallocate non-paged pool
              0E   BB   0D4E          PUSHR   #^M<R1,R2,R3>             ; Save regs
              7E   D4   0D50          CLRL    -(SP)                     ; Value to return in R0
                        0D52                                            ;
        51    08 A0  3C   0D52          MOVZWL  8(R0),R1                  ; Get size of block
   00000000'GF     16   0D56          JSB     G^EXE$DEANONPGDSIZ        ; Deallocate it
                        0D5C                                            ;
              0F   BA   0D5C          POPR    #^M<R0,R1,R2,R3>          ; Restore regs
                   05   0D5E          RSB                               ; Done
                        0D5F
                        0D5F  .SBTTL  NET$MOV_TO_XWB  - Move counted string to XWB$B_DATA
                        0D5F  .SBTTL  NET$MOV_CSTR    - Move counted string with count field
                        0D5F  .SBTTL  NET$MOV_USTR    - Move counted string without count field
                        0D5F  ;+
                        0D5F  ;
                        0D5F  ;  The source string is moved to its destination.  Both the source
                        0D5F  ;  and destination pointers are updated.
                        0D5F  ;
                        0D5F  ;
                        0D5F  ;  INPUTS:        R3      Pointer to destination field
                        0D5F  ;                 R1      Pointer to count field of source string
                        0D5F  ;
                        0D5F  ;  OUTPUTS:       R3      Pointer to first byte beyond end of destination
                        0D5F  ;                 R1      Pointer to first byte beyond source string
                        0D5F  ;
                        0D5F  ;                 All other registers are preserved
                        0D5F  ;
                        0D5F  ;
                        0D5F  ;-
                        0D5F                    .ENABL  LSB
                        0D5F  NET$MOV_TO_XWB::                                  ; Move counted string to XWB$B_DATA
                53   DD 0D5F          PUSHL     R3                              ; Save reg
       53   5B  A5   9E 0D61          MOVAB     XWB$B_DATA(R5),R3               ; Setup destination ptr
                04   10 0D65          BSBB      NET$MOV_CSTR                    ; Move the string
                53 8ED0 0D67          POPL      R3                              ; Restore reg
                     05 0D6A          RSB                                       ; Done
                        0D6B                                                    ;
                        0D6B  NET$MOV_CSTR::                                    ; Move counted string with count byte
                35   BB 0D6B          PUSHR     #^M<R0,R2,R4,R5>                ; Save regs
                        0D6D
            50  61   9B 0D6D          MOVZBW    (R1),R0                         ; Get string length
                50   B6 0D70          INCW      R0                              ; Include count itself
                05   11 0D72          BRB       10$                             ; Continue in common
                        0D74                                                    ;
                        0D74  NET$MOV_USTR::                                    ; Mov counted str w/o count byte
                35   BB 0D74          PUSHR     #^M<R0,R2,R4,R5>                ; Save regs
                        0D76
            50  81   9B 0D76          MOVZBW    (R1)+,R0                        ; Get count value, advance ptr
       63   61  50   28 0D79  10$:    MOVC3     R0,(R1),(R3)                    ; Move the string
                        0D7D                                                    ;
                35   BA 0D7D          POPR      #^M<R0,R2,R4,R5>                ; Restore regs
                     05 0D7F          RSB
                        0D80
                        0D80                    .DSABL LSB
                        0D80
                        0D80  .SBTTL  NET$POST_IO     - Send IRP to COM$POST
                        0D80  ;+
                        0D80  ;
                        0D80  ;  INPUTS:       R3      IRP address
                        0D80  ;                R0      Scratch
                        0D80  ;
                        0D80  ;  OUTPUTS:      R0      SS$_NORMAL
                        0D80  ;
                        0D80  ;                All other registers are preserved
                        0D80  ;-
                        0D80  NET$POST_IO::                                   ; Send IRP to COM$POST
                        0D80          ;
                        0D80          ;
                        0D80          ;  Complete the I/O
                        0D80          ;
                        0D80          ;
              55     DD 0D80          PUSHL   R5                              ; Save XWB pointer
     55    1C A3     D0 0D82          MOVL    IRP$L_UCB(R3),R5                ; Get UCB address
   00000000'GF       16 0D86          JSB     G^COM$POST                      ; Another packet for the heap
        50    01     D0 0D8C          MOVL    S^#SS$_NORMAL,R0                ; Always return success
              55   8ED0 0D8F          POPL    R5                              ; Recover XWB address
                     05 0D92          RSB                                     ; Done
                        0D93
                        0D93
                        0D93
                        0D93
                        0D93  .END
```

```
$$$                             = 00000020 R       02    ACT$_RCV_DATA                    = 00000009
$$OP                            = 00000002               ACT$_RCV_DTACK                   = 0000000A
$$_NSPMSG                       = 00000000               ACT$_RCV_DX                      = 0000000D
$$_TR3MSG                       = 00000000               ACT$_RCV_LI                      = 0000000B
$$_TR4MSG                       = 00000000               ACT$_RCV_LIACK                   = 0000000C
ABORT                             000009AB R       03    ACT$_RCV_RTS                     = 00000008
ABORT_ALL                         00000A08 R       03    ACT$_RES_DISC                    = 00000010
ACB$B_RMOD                      = 0000000B               ACT$_RTS_NLT                     = 00000006
ACB$C_LENGTH                    = 0000001C               ACT$_SHRLNK                      = 00000014
ACB$L_KAST                      = 00000018               ACT$_SSABORT                     = 00000012
ACB$L_PID                       = 0000000C               ACT_DISPATCH                       00000447 R       03
ACP$ACCESSNET                     ********    X    03    AQB$L_ACPPID                     = 0000000C
ACP$C_STA_F                     = 00000004               AQB$L_ACPQBL                     = 00000004
ACP$C_STA_H                     = 00000005               AT$_NULL                           ********    X    02
ACP$C_STA_I                     = 00000000               BIT...                           = 00000004
ACP$C_STA_N                     = 00000001               BRDCST                             00000A53 R       03
ACP$C_STA_R                     = 00000002               BUG$_NETNOSTATE                    ********    X    03
ACP$C_STA_S                     = 00000003               CHANGE_STA                         0000036F R       03
ACP$DEACCESS                      ********    X    03    CHKRETADDR                         00000345 R       03
ACP$MODIFY                        ********    X    03    CHK_X_IRP                          000007D1 R       03
ACT$ABORT                         ********    X    03    CLEANUP_ACCESS                     00000814 R       03
ACT$BUG                           0000047A R       03    CNF$_ADVANCE                     = 00000000
ACT$CANLNK                        ********    X    03    CNF$_QUIT                        = 00000002
ACT$CONFIRM                       00000618 RG      03    CNF$_TAKE_CURR                   = 00000003
ACT$DEACCESS                      0000078C RG      03    CNF$_TAKE_PREV                   = 00000001
ACT$ENT_RUN                       0000070B RG      03    COM$POST                           ********    X    03
ACT$INITIATE                      0000064F RG      03    CRB$L_INTD                       = 00000024
ACT$LOG                           0000047E R       03    CRE_LNK                            000009DF R       03
ACT$NOLINK                        00000484 R       03    CXB$B_R_AREA                       00000039
ACT$NOP                           00000479 R       03    CXB$B_R_FLG                        00000038
ACT$RCV_CA                        ********    X    03    CXB$B_R_NSPTYP                     00000039
ACT$RCV_CC                        ********    X    03    CXB$B_TYPE                       = 0000000A
ACT$RCV_CI                        ********    X    03    CXB$B_X_NSPTYP                     0000004E
ACT$RCV_CR                        ********    X    03    CXB$C_DCL                        = 00000020
ACT$RCV_DATA                      ********    X    03    CXB$C_HEADER                     = 00000048
ACT$RCV_DTACK                     ********    X    03    CXB$C_R_LENGTH                   = 0000003C
ACT$RCV_DX                        ********    X    03    CXB$L_LINK                       = 00000010
ACT$RCV_LI                        ********    X    03    CXB$L_R_MSG                        0000002C
ACT$RCV_LIACK                     ********    X    03    CXB$L_R_RCB                        00000028
ACT$RCV_RTS                       ********    X    03    CXB$T_DCL                        = 00000028
ACT$RES_DISC                      000007B2 RG      03    CXB$T_X_DATA                       00000057
ACT$RTS_NLT                       ********    X    03    CXB$T_X_XPORT                      00000048
ACT$SHRLNK                        0000048B R       03    CXB$W_R_ADJ                        0000003A
ACT$SSABORT                       0000048B R       03    CXB$W_R_BCNT                       00000030
ACT$_ABORT                      = 0000000E               CXB$W_R_DSTNOD                     00000034
ACT$_BUG                        = 00000000  G            CXB$W_R_NSPSEQ                     0000003A
ACT$_CANLNK                     = 0000000F               CXB$W_R_PATH                       00000032
ACT$_CONFIRM                    = 00000013               CXB$W_R_SRCNOD                     00000036
ACT$_DEACCESS                   = 00000015               CXB$W_SIZE                       = 00000008
ACT$_ENT_RUN                    = 00000007               CXB$W_X_NSPACK                     00000053
ACT$_INITIATE                   = 00000011               CXB$W_X_NSPLOC                     00000051
ACT$_LOG                        = 00000000               CXB$W_X_NSPREM                     0000004F
ACT$_NOP                        = 00000004               CXB$W_X_NSPSEQ                     00000055
ACT$_RCV_CA                     = 00000003               DDB$L_DDT                        = 0000000C
ACT$_RCV_CC                     = 00000005               DDB$T_NAME                       = 00000014
ACT$_RCV_CI                     = 00000002               DEAL_ICB                           00000861 R       03
ACT$_RCV_CR                     = 00000001               DECLARE                            00000AA5 R       03
```

```
DEV$M_AVL                       ********   X  02      IO$_ACPCONTROL                = 00000038
DEV$M_IDV                       ********   X  02      IO$_DEACCESS                  = 00000034
DEV$M_MBX                       ********   X  02      IO$_READLBLK                  = 00000021
DEV$M_NET                       ********   X  02      IO$_READVBLK                  = 00000031
DEV$M_ODV                       ********   X  02      IO$_SETMODE                   = 00000023
DISC_ONE                        000009ED R    03      IO$_VIRTUAL                   = 0000003F
DLLTRN                          00000AD8 R    03      IO$_WRITELBLK                 = 00000020
DPT$B_FLAGS                   = 0000000D              IO$_WRITEVBLK                 = 00000030
DPT$C_LENGTH                  = 00000038              IOC$INITIATE                    ********   X  03
DPT$C_VERSION                 = 00000004              IOC$MNTVER                      ********   X  03
DPT$INITAB                      00000038 R    02      IOC$REQCOM                      ********   X  03
DPT$M_NOUNLOAD                = 00000004              IOC$RETURN                      ********   X  03
DPT$REINITAB                    00000074 R    02      IPL$_SYNCH                    = 00000008
DPT$TAB                         00000000 R    02      IPL$_TIMER                    = 00000008
DRAIN_RCV                       00000921 R    03      IRP$C_BCNT                    = 00000032
DRAIN_XMT                       000008C2 R    03      IRP$L_DIAGBUF                 = 0000004C
DYN$C_CRB                     = 00000005              IRP$L_IOQFL                   = 00000000
DYN$C_CXB                     = 0000001B              IRP$L_IOST1                   = 00000038
DYN$C_DDB                     = 00000006              IRP$L_PID                     = 0000000C
DYN$C_DPT                     = 0000001E              IRP$L_SVAPTE                  = 0000002C
DYN$C_NDB                     = 0000001C              IRP$L_UCB                     = 0000001C
DYN$C_ORB                     = 00000049              IRP$L_WIND                    = 00000018
DYN$C_UCB                     = 00000010              IRP$M_FUNC                    = 00000002
EXE$ABORTIO                     ********   X  03      IRP$Q_NT_PRVMSK               = 00000040
EXE$ALONONPAGED                 ********   X  03      IRP$V_COMPLX                  = 00000003
EXE$DEANONPGDSIZ                ********   X  03      IRP$W_BCNT                    = 00000032
EXE$FINISHIO                    ********   X  03      IRP$W_CHAN                    = 00000028
EXE$FORK                        ********   X  03      IRP$W_FUNC                    = 00000020
EXE$GQ_SYSTIME                  ********   X  03      IRP$W_STS                     = 0000002A
EXE$INSTIMQ                     ********   X  03      JIB$L_BYTCNT                  = 00000020
EXE$WRTMAILBOX                  ********   X  03      JIB$L_BYTLM                   = 00000024
EXIT                            000009BA R    03      JIB$W_FILCNT                  = 00000030
FKB$C_LENGTH                  = 00000018              LPD$B_PTH_INX                 = 00000020
FUNCTABLE                       00000038 R    03      LPD$C_LOC_INX                 = 00000001
FUNCTAB_LEN                   = 00000058              LSB                           = 00000000
GET_P1DSC                       000007F2 R    03      LSB$B_R_CXBCNT                = 00000028
GET_P2DSC                       000007F7 R    03      LSB$B_R_CXBQUO                = 00000029
GET_P3DSC                       000007FC R    03      LSB$B_SPARE                   = 0000002A
GET_P4DSC                       00000801 R    03      LSB$B_STS                     = 0000002B
GET_WNDSC                       000007EE R    03      LSB$B_X_ADJ                   = 0000000B
ICB$B_DATA                    = 0000007C              LSB$B_X_CXBACT                = 0000000D
ICB$B_RID                     = 00000092              LSB$B_X_CXBCNT                = 0000000F
ICB$C_RID                     = 00000010              LSB$B_X_CXBQUO                = 0000000E
ICB$T_RID                     = 00000093              LSB$B_X_PKTWND                = 0000000C
ICB$W_DLY_FACT                = 0000000E              LSB$B_X_REQ                   = 0000000A
ICB$W_DLY_WGHT                = 00000010              LSB$L_CROSS                   = 0000002C
ICB$W_LOCLNK                  = 00000002              LSB$L_R_CXB                   = 00000020
ICB$W_PATH                    = 00000000              LSB$L_R_IRP                   = 0000001C
ICB$W_RETRAN                  = 0000000C              LSB$L_X_CXB                   = 00000018
ICB$W_SEGSIZ                  = 00000012              LSB$L_X_IRP                   = 00000014
ICB$W_TIM_INACT               = 00000006              LSB$L_X_PND                   = 00000010
ICB$W_TIM_OCON                = 00000004              LSB$M_BOM                     = 00000020
INIT_XWB                        00000C1F R    03      LSB$M_EOM                     = 00000040
IO$M_FCODE                    = 0000003F              LSB$M_LI                      = 00000001
IO$M_INTERRUPT                = 00000040              LSB$S_LSB                     = 00000030
IO$V_ABORT                    = 00000008              LSB$S_SPARE                   = 00000004
IO$_ACCESS                    = 00000032              LSB$S_STS                     = 00000001
```

```
LSB$V_BOM                     = 00000005                NET$C_DR_ACCESS               = 00000022
LSB$V_EOM                     = 00000006                NET$C_DR_BUSY                 = 00000006
LSB$V_LI                      = 00000000                NET$C_DR_DEACC                = 00000066  G
LSB$V_SPARE                   = 00000001                NET$C_DR_EXIT                 = 00000026
LSB$W_HAA                     = 00000008                NET$C_DR_FMT                  = 00000005
LSB$W_HAR                     = 00000006                NET$C_DR_INVALID              = 00000064  G
LSB$W_HAX                     = 00000026                NET$C_DR_IVNODE               = 00000002
LSB$W_HNR                     = 00000024                NET$C_DR_NOBJ                 = 00000004
LSB$W_HXS                     = 00000004                NET$C_DR_NONODE               = 00000002
LSB$W_LNX                     = 00000002                NET$C_DR_NOPATH               = 00000027
LSB$W_LUX                     = 00000000                NET$C_DR_NORMAL               = 00000000
LTB$L_SLOTS                   = 00000010                NET$C_DR_RSU                  = 00000001
LTB$L_XWB                     = 0000000C                NET$C_DR_SHUT                 = 00000003
LTB$W_SLT_TOT                 = 00000004                NET$C_DR_THIRD                = 00000008
MASKH                         = 01000000                NET$C_EFN_ASYN                = 00000002
MASKL                         = 00000000                NET$C_EFN_WAIT                = 00000001
MBX$M_EVTAVL                  = 00000002                NET$C_IPL                     = 00000008
MBX$M_EVTRCVCHG               = 00000004                NET$C_MAXACCFLD               = 00000027
MBX$M_EVTXMTCHG               = 00000008                NET$C_MAXLINNAM               = 0000000F
MBX$M_NETSTATE                = 00000001                NET$C_MAXLNK                  = 000003FF
MBX$V_EVTAVL                  = 00000001                NET$C_MAXNODNAM               = 00000006
MBX$V_EVTRCVCHG               = 00000002                NET$C_MAXOBJNAM               = 0000000C
MBX$V_EVTXMTCHG               = 00000003                NET$C_MAX_AREAS               = 0000003F
MBX$V_NETSTATE                = 00000000                NET$C_MAX_LINES               = 00000040
MBX_TABLE                       000001E8 R       03     NET$C_MAX_NCB                 = 0000006E
MSG$_ABORT                    = 00000030                NET$C_MAX_NODES               = 000003FF
MSG$_CONNECT                  = 00000032                NET$C_MAX_OBJ                 = 000000FF
MSG$_DISCON                   = 00000033                NET$C_MAX_WQE                 = 00000014
MSG$_EVTAVL                   = 0000003E                NET$C_MINBUFSIZ               = 000000C0
MSG$_EVTRCVCHG                = 0000003F                NET$C_STABITS                 = 00000003
MSG$_EVTXMTCHG                = 00000044                NET$C_TID_ACT                 = 00000003
MSG$_EXIT                     = 00000034                NET$C_TID_RUS                 = 00000001
MSG$_NETSHUT                  = 0000003B                NET$C_TID_XRT                 = 00000002
MSG$_PATHLOST                 = 00000036                NET$C_TRCTL_CEL               = 00000002
MSG$_REJECT                   = 00000038                NET$C_TRCTL_OVR               = 00000005
MSG$_THIRDPARTY               = 00000039                NET$C_UTLBUFSIZ               = 00001000
NET$AB_STTAB                    0000013C R       03     NET$DDT                         00000000 RG      03
NET$ACCESS                      000005B1 R       03     NET$DEACCESS                    00000777 RG      03
NET$ACK_XMT_SEGS                ********    X    03     NET$DEALLOCATE                  00000D4E RG      03
NET$ACP_COMM                    00000969 RG      03     NET$DRAIN_FREE_CXB              00000D16 RG      03
NET$ALONONPAGED                 00000D34 RG      03     NET$END                         ********    X    02
NET$ALONPGD_Z                   00000D23 RG      03     NET$END_EVENT                   0000032A RG      03
NET$ALTENTRY                    ********    X    03     NET$EVENT                       00000356 RG      03
NET$AW_FLG_CLRM                 0000012C R       03     NET$FDT_ACCESS                  000005A3 R       03
NET$AW_FLG_SETM                 0000011C R       03     NET$FDT_CONTROL                 00000538 R       03
NET$AZ_DR_CONTAB                00000262 R       03     NET$FDT_DEACCESS                00000711 RG      03
NET$AZ_DR_TABLE                 00000204 R       03     NET$FDT_RCV                     ********    X    03
NET$CANCEL                      0000087B R       03     NET$FDT_SETMODE                 0000050B R       03
NET$CHK_X_IDLE                  000007C2 RG      03     NET$FDT_XMT                     ********    X    03
NET$CMPL_ACC                    000006CA RG      03     NET$FORK                        000002EC RG      03
NET$COMPLEX_EV                  00000330 RG      03     NET$GL_OFF_DPTFLG               00000118 RG      03
NET$CONTROL                     0000054A R       03     NET$GL_WORKBITS                 000001E4 RG      03
NET$CREATE_XWB                  00000BC6 RG      03     NET$GQ_PATCH                    00000090 RG      03
NET$CTLR_INIT                   000002E1 R       03     NET$INTERRUPT                   000002E1 R       03
NET$C_ACTBITS                 = 00000005                NET$KAST                        ********    X    03
NET$C_ACT_TIMER               = 0000001E                NET$MAP_R_REASON                000002C0 RG      03
NET$C_DR_ABORT                = 00000009                NET$MARK_LINK                   ********    X    03
```

```
NET$MOV_CSTR                      00000D6B RG     03     NSP$$$_QUAL_FLW                 = 00000000
NET$MOV_TO_XWB                    00000D5F RG     03     NSP$$$_QUAL_INF                 = 00000000
NET$MOV_USTR                      00000D74 RG     03     NSP$$$_QUAL_MSG                 = 00000000
NET$M_MAXLNKMSK                 = 000003FF               NSP$$$_QUAL_SRV                 = 00000000
NET$M_STAMSK                    = 000000E0               NSP$C_EXT_LNK                   = 0000001E
NET$POST_IO                       00000D80 RG     03     NSP$C_FLW_DATA                  = 00000000
NET$PRE_EMPT                      00000341 RG     03     NSP$C_FLW_INT                   = 00000001
NET$PURG_RUN                      000008AF RG     03     NSP$C_FLW_NOP                   = 00000000
NET$QUE_XWB                       00000CF1 RG     03     NSP$C_FLW_XOFF                  = 00000001
NET$RCV_DONE                      ********    X   03     NSP$C_FLW_XON                   = 00000002
NET$RESET_TIMER                   ********    X   03     NSP$C_HSZ_ACK                   = 00000007
NET$RET_SLOT                      00000CD1 RG     03     NSP$C_HSZ_CA                    = 00000003
NET$SCH_MSG                       000003A3 RG     03     NSP$C_HSZ_CC                    = 00000064
NET$SEND_CS_MBX                   00000B31 RG     03     NSP$C_HSZ_CD                    = 000000F0
NET$SEND_MBX                      00000B71 RG     03     NSP$C_HSZ_CI                    = 000000F0
NET$SETUP_RUN                     ********    X   03     NSP$C_HSZ_DATA                  = 00000009
NET$STARTIO                       00000490 R      03     NSP$C_HSZ_DC                    = 00000016
NET$TIMER                         ********    X   03     NSP$C_HSZ_DI                    = 00000016
NET$UNIT_INIT                     000002E2 R      03     NSP$C_HSZ_INT                   = 00000009
NET$UNSOL_INTR                    ********    X   03     NSP$C_HSZ_LS                    = 00000009
NET$XMT_DONE                      ********    X   03     NSP$C_INF_V31                   = 00000001
NET$XWB_LOCLNK                    00000CA9 RG     03     NSP$C_INF_V32                   = 00000000
NETEVT$_CA                      = 00000001  G            NSP$C_INF_V33                   = 00000002
NETEVT$_CANLNK                  = 0000000D  G            NSP$C_MAXHDR                    = 00000009
NETEVT$_CC                      = 00000002  G            NSP$C_MAX_DELAY                 = 00000014
NETEVT$_CCA                     = 00000010  G            NSP$C_MAX_R_CXB                 = 00000007
NETEVT$_CI                      = 00000000  G            NSP$C_MAX_XPW                   = 00000007
NETEVT$_CIA                     = 0000000F  G            NSP$C_MSG_CA                    = 00000024
NETEVT$_CRA                     = 00000011  G            NSP$C_MSG_CC                    = 00000028
NETEVT$_DATA                    = 00000005  G            NSP$C_MSG_CI                    = 00000018
NETEVT$_DC                      = 0000000B  G            NSP$C_MSG_DATA                  = 00000000
NETEVT$_DEA                     = 00000012  G            NSP$C_MSG_DC                    = 00000048
NETEVT$_DI                      = 0000000A  G            NSP$C_MSG_DI                    = 00000038
NETEVT$_DSCLNK                  = 0000000C  G            NSP$C_MSG_DTACK                 = 00000004
NETEVT$_DTACK                   = 00000006  G            NSP$C_MSG_INT                   = 00000030
NETEVT$_INT                     = 00000008  G            NSP$C_MSG_LIACK                 = 00000014
NETEVT$_LIACK                   = 00000009  G            NSP$C_MSG_LS                    = 00000010
NETEVT$_LS                      = 00000007  G            NSP$C_SRV_MFC                   = 00000002
NETEVT$_MBXERR                  = 00000013  G            NSP$C_SRV_NFC                   = 00000000
NETEVT$_PH2CCS                  = 00000003  G            NSP$C_SRV_REQ                   = 00000001
NETEVT$_PROERR                  = 00000014  G            NSP$C_SRV_SFC                   = 00000001
NETEVT$_RESDIS                  = 0000000E  G            NSP$M_ACK_NAK                   = 00001000
NETEVT$_RTS                     = 00000004  G            NSP$M_ACK_NUM                   = 00000FFF
NETUPD$_ABOLNK                  = 00000008               NSP$M_ACK_VALID                 = 00008000
NETUPD$_ABORT                   = 00000001               NSP$M_DATA_BOM                  = 00000020
NETUPD$_BRDCST                  = 0000000A               NSP$M_DATA_EOM                  = 00000040
NETUPD$_CONNECT                 = 00000002               NSP$M_DATA_OVFW                 = 00000080
NETUPD$_CRELNK                  = 00000007               NSP$M_FLW_CHAN                  = 0000000C
NETUPD$_DLL_ON                  = 00000005               NSP$M_FLW_DRV                   = 000000F0
NETUPD$_DSCLNK                  = 00000009               NSP$M_FLW_INT                   = 00000020
NETUPD$_EXIT                    = 00000003               NSP$M_FLW_INUSE                 = 00000010
NETUPD$_PROCRE                  = 00000004               NSP$M_FLW_LISUB                 = 00000004
NETUPD$_REPLY                   = 0000000B               NSP$M_FLW_MODE                  = 00000003
NEW_STATE                         0000037E R      03     NSP$M_FLW_SP1                   = 00000008
NSP$$$_QUAL_ACK                 = 00000000               NSP$M_FLW_SP2                   = 00000040
NSP$$$_QUAL_ALTFLW              = 00000000               NSP$M_FLW_SP3                   = 00000080
NSP$$$_QUAL_DATA                = 00000000               NSP$M_FLW_XOFF                  = 00000001
```

```
NSP$M_FLW_XON                    = 00000002                     NSP$V_SRV_SP1                    = 00000004
NSP$M_INF_VER                    = 00000003                     NSP$W_DSTLNK                     = 00000001
NSP$M_MSG_INT                    = 00000020                     NSP$W_SRCLNK                     = 00000003
NSP$M_MSG_LI                     = 00000010                     ORB$B_FLAGS                      = 00000008
NSP$M_SRV_01                     = 00000003                     ORB$L_OWNER                      = 00000000
NSP$M_SRV_EXT                    = 00000080                     ORB$M_PROT_16                    = 00000001
NSP$M_SRV_FLW                    = 0000000C                     ORB$W_PROT                       = 00000018
NSP$M_SRV_REQ                    = 000000F3                     P1                               = 00000000
NSP$M_SRV_SP1                    = 00000070                     P2                               = 00000004
NSP$R_QUAL                       = 00000000                     P3                               = 00000008
NSP$SOLICIT                        ********    X    03          PATCH_AREA_SIZE                  = 00000080
NSP$S_ACK_NUM                    = 0000000C                     PCB$L_JIB                        = 00000080
NSP$S_ACK_SP2                    = 00000002                     PCB$L_PHD                        = 0000006C
NSP$S_DATA_SP                    = 00000005                     PCB$L_PID                        = 00000060
NSP$S_FLW_CHAN                   = 00000002                     PHD$Q_PRIVMSK                    = 00000000
NSP$S_FLW_DRV                    = 00000004                     PR$_IPL                          = 00000012
NSP$S_FLW_MODE                   = 00000002                     PROCRE                             0000099E R       03
NSP$S_INF_VER                    = 00000002                     PROC_IO                            000004DA R       03
NSP$S_MSG_SP1                    = 00000004                     R3_OFF                           = 0000000C
NSP$S_NSPMSG                     = 00000005                     R4_OFF                           = 00000010
NSP$S_QUAL                       = 00000005                     R5_OFF                           = 00000014
NSP$S_QUAL_ACK                   = 00000002                     RCB$B_ECL_DFA                    = 00000064
NSP$S_QUAL_ALTFLW                = 00000001                     RCB$B_ECL_DWE                    = 00000065
NSP$S_QUAL_DATA                  = 00000001                     RCB$B_ECL_RFA                    = 00000063
NSP$S_QUAL_FLW                   = 00000001                     RCB$L_ACP_UCB                    = 00000014
NSP$S_QUAL_INF                   = 00000001                     RCB$L_AQB                        = 00000010
NSP$S_QUAL_MSG                   = 00000005                     RCB$L_PTR_LTB                    = 00000024
NSP$S_QUAL_SRV                   = 00000001                     RCB$L_PTR_TQE                    = 00000030
NSP$S_SRV_01                     = 00000002                     RCB$W_ADDR                       = 0000000E
NSP$S_SRV_FLW                    = 00000002                     RCB$W_CNT_MLL                    = 0000009E
NSP$S_SRV_SP1                    = 00000003                     RCB$W_CNT_XRE                    = 0000009C
NSP$V_ACK_NAK                    = 0000000C                     RCB$W_ECLSEGSIZ                  = 0000007C
NSP$V_ACK_NUM                    = 00000000                     RCB$W_MAX_LNK                    = 00000058
NSP$V_ACK_SP2                    = 0000000D                     RCB$W_MCOUNT                     = 00000054
NSP$V_ACK_VALID                  = 0000000F                     RCB$W_TIM_CNI                    = 00000076
NSP$V_DATA_BOM                   = 00000005                     RCB$W_TRANS                      = 0000000C
NSP$V_DATA_EOM                   = 00000006                     REASON_C_LENGTH                  = 00000006   G
NSP$V_DATA_OVFW                  = 00000007                     REASON_W_DR                      = 00000000   G
NSP$V_DATA_SP                    = 00000000                     REASON_W_MBX                     = 00000004   G
NSP$V_FLW_CHAN                   = 00000002                     REASON_W_SS                      = 00000002   G
NSP$V_FLW_DRV                    = 00000004                     REPLY                              00000AA0 R       03
NSP$V_FLW_INT                    = 00000005                     SCH$GL_PCBVEC                      ********    X    03
NSP$V_FLW_INUSE                  = 00000004                     SCH$WAKE                           ********    X    03
NSP$V_FLW_LISUB                  = 00000002                     SETUP_XWB                          0000064F R       03
NSP$V_FLW_MODE                   = 00000000                     SIZ...                           = 00000001
NSP$V_FLW_SP1                    = 00000003                     SS$_ABORT                        = 0000002C
NSP$V_FLW_SP2                    = 00000006                     SS$_ACCVIO                       = 0000000C
NSP$V_FLW_SP3                    = 00000007                     SS$_BADPARAM                     = 00000014
NSP$V_FLW_XOFF                   = 00000000                     SS$_CONNECFAIL                   = 000020DC
NSP$V_FLW_XON                    = 00000001                     SS$_DEVALLOC                     = 00000840
NSP$V_INF_VER                    = 00000000                     SS$_FILNOTACC                    = 000000AC
NSP$V_MSG_INT                    = 00000005                     SS$_ILLIOFUNC                    = 000000F4
NSP$V_MSG_LI                     = 00000004                     SS$_INVLOGIN                     = 0000209C
NSP$V_MSG_SP1                    = 00000000                     SS$_LINKABORT                    = 000020E4
NSP$V_SRV_01                     = 00000000                     SS$_LINKDISCON                   = 000020EC
NSP$V_SRV_EXT                    = 00000007                     SS$_LINKEXIT                     = 000020F4
NSP$V_SRV_FLW                    = 00000002                     SS$_NOLINKS                      = 0000027C
```

```
SS$_NOMBX                    = 00000274                 TR4$$$_QUAL_RTFLG            = 00000000
SS$_NORMAL                   = 00000001                 TR4$$$_QUAL_SCLASS           = 00000000
SS$_NOSUCHNODE               = 0000028C                 TR4$C_BCE_MID1               = 040000AB
SS$_NOSUCHOBJ                = 000020A4                 TR4$C_BCE_MID2               = 00000000
SS$_PATHLOST                 = 000020FC                 TR4$C_BCR_MID1               = 030000AB
SS$_PROTOCOL                 = 00002074                 TR4$C_BCR_MID2               = 00000000
SS$_REJECT                   = 00000294                 TR4$C_BCT3MULT               = 00000008
SS$_REMRSRC                  = 0000206C                 TR4$C_END_NODE               = 00000003
SS$_SHUT                     = 0000208C                 TR4$C_HIORD                  = 000400AA
SS$_THIRDPARTY               = 0000207C                 TR4$C_HSZ_DATA               = 00000015
SS$_UNREACHABLE              = 00002094                 TR4$C_MSG_BCEHEL             = 0000000D
TQE$B_RQTYPE                 = 0000000B                 TR4$C_MSG_BCRHEL             = 0000000B
TQE$C_SSREPT                 = 00000005                 TR4$C_MSG_LDATA              = 00000006
TQE$L_FPC                    = 0000000C                 TR4$C_MSG_RDATA              = 00000002
TQE$L_FR4                    = 00000014                 TR4$C_PRO_TYPE               = 00000360
TQE$Q_DELTA                  = 00000020                 TR4$C_RTR_LVL1               = 00000002
TQE$V_REPEAT                 = 00000002                 TR4$C_RTR_LVL2               = 00000001
TR$C_MAXHDR                  = 0000001C                 TR4$C_T3MULT                 = 00000002
TR$C_NI_ALLEND1              = 040000AB                 TR4$C_VER_HIB                = 00000000
TR$C_NI_ALLEND2              = 00000000                 TR4$C_VER_LOWW               = 00000002
TR$C_NI_ALLROU1              = 030000AB                 TR4$M_ADDR_AREA              = 0000FC00
TR$C_NI_ALLROU2              = 00000000                 TR4$M_ADDR_DEST              = 000003FF
TR$C_NI_PREFIX               = 000400AA                 TR4$M_RTFLG_INI              = 00000020
TR$C_NI_PROT                 = 00000360                 TR4$M_RTFLG_LNG              = 00000004
TR$C_PRI_ECL                 = 0000001F                 TR4$M_RTFLG_RQR              = 00000008
TR$C_PRI_RTHRU               = 0000001F                 TR4$M_RTFLG_RTS              = 00000010
TR$UPDATE                      ********   X    03       TR4$R_QUAL                   = 00000000
TR3$$$_QUAL_MSG              = 00000000                 TR4$S_ADDR_AREA              = 00000006
TR3$$$_QUAL_RTFLG            = 00000000                 TR4$S_ADDR_DEST              = 0000000A
TR3$C_HSZ_DATA               = 00000006                 TR4$S_QUAL                   = 00000002
TR3$C_MSG_DATA               = 00000002                 TR4$S_QUAL_ADDR              = 00000002
TR3$C_MSG_HELLO              = 00000005                 TR4$S_QUAL_RTFLG             = 00000001
TR3$C_MSG_INIT               = 00000001                 TR4$S_QUAL_SCLASS            = 00000001
TR3$C_MSG_NOP2               = 00000008                 TR4$S_RTFLG_01               = 00000002
TR3$C_MSG_ROUT               = 00000007                 TR4$S_RTFLG_VER              = 00000002
TR3$C_MSG_STR2               = 00000058                 TR4$S_SCLASS_57              = 00000003
TR3$C_MSG_VERF               = 00000003                 TR4$S_TR4MSG                 = 00000002
TR3$M_MSG_CTL                = 00000001                 TR4$V_ADDR_AREA              = 0000000A
TR3$M_MSG_RTH                = 00000002                 TR4$V_ADDR_DEST              = 00000000
TR3$M_RTFLG_PH2              = 00000040                 TR4$V_RTFLG_01               = 00000000
TR3$M_RTFLG_RQR              = 00000008                 TR4$V_RTFLG_INI              = 00000005
TR3$M_RTFLG_RTS              = 00000010                 TR4$V_RTFLG_LNG              = 00000002
TR3$R_QUAL                   = 00000000                 TR4$V_RTFLG_RQR              = 00000003
TR3$S_QUAL                   = 00000001                 TR4$V_RTFLG_RTS              = 00000004
TR3$S_QUAL_MSG               = 00000001                 TR4$V_RTFLG_VER              = 00000006
TR3$S_QUAL_RTFLG             = 00000001                 TR4$V_SCLASS_1               = 00000001
TR3$S_RTFLG_012              = 00000003                 TR4$V_SCLASS_57              = 00000005
TR3$S_TR3MSG                 = 00000001                 TR4$V_SCLASS_BC              = 00000004
TR3$V_MSG_CTL                = 00000000                 TR4$V_SCLASS_LS              = 00000002
TR3$V_MSG_RTH                = 00000001                 TR4$V_SCLASS_METR            = 00000000
TR3$V_RTFLG_012              = 00000000                 TR4$V_SCLASS_SUBA            = 00000003
TR3$V_RTFLG_5                = 00000005                 UCB$B_DIPL                   = 0000005E
TR3$V_RTFLG_7                = 00000007                 UCB$B_FIPL                   = 0000000B
TR3$V_RTFLG_PH2              = 00000006                 UCB$C_LENGTH                 = 00000090
TR3$V_RTFLG_RQR              = 00000003                 UCB$L_AMB                    = 00000060
TR3$V_RTFLG_RTS              = 00000004                 UCB$L_DDB                    = 00000028
TR4$$$_QUAL_ADDR             = 00000000                 UCB$L_DEVCHAR                = 00000038
```

```
UCB$L_DEVDEPEND                 = 00000044              XWB$L_LINK                      = 0000002C
UCB$L_IOQFL                     = 0000004C              XWB$L_ORGUCB                    = 00000010
UCB$L_IRP                       = 00000058              XWB$L_PID                       = 00000034
UCB$L_LINK                      = 00000030              XWB$L_PTR_RTHD                  = 00000120   G
UCB$L_VCB                       = 00000034              XWB$L_VCB                       = 00000030
UCB$M_BSY                       = 00000100              XWB$L_WLBL                      = 00000004
UCB$M_ONLINE                    = 00000010              XWB$L_WLFL                      = 00000000
UCB$M_TEMPLATE                  = 00002000              XWB$M_FLG_BREAK                 = 00000001
UCB$V_BSY                       = 00000008              XWB$M_FLG_CLO                   = 00000200
UCB$W_DEVBUFSIZ                 = 00000042              XWB$M_FLG_IAVL                  = 00001000
UCB$W_MB_SEED                   = 00000000              XWB$M_FLG_SCD                   = 00000100
UCB$W_STS                       = 00000064              XWB$M_FLG_SDACK                 = 00000008
UCB$W_UNIT                      = 00000054              XWB$M_FLG_SDFL                  = 00004000
UNKNOWN                           00000B2D R      03    XWB$M_FLG_SDT                   = 00000080
VEC$L_ADP                       = 00000014              XWB$M_FLG_SIACK                 = 00000004
VEC$L_INITIAL                   = 0000000C              XWB$M_FLG_SIFL                  = 00002000
VEC$L_START                     = 0000001C              XWB$M_FLG_SLI                   = 00000010
VEC$L_UNITINIT                  = 00000018              XWB$M_FLG_TBPR                  = 00000800
XWB                             = 00000000              XWB$M_FLG_WBP                   = 00000040
XWB$$                           = 00000160   G          XWB$M_FLG_WBUF                  = 00000002
XWB$B_ACCESS                    = 0000000B              XWB$M_FLG_WDAT                  = 00000400
XWB$B_ADJ_INX                   = 00000124   G          XWB$M_FLG_WHGL                  = 00000020
XWB$B_DATA                      = 0000005B              XWB$M_FLG_WMSK                  = 0000039D
XWB$B_FIPL                      = 0000001F              XWB$M_PRO_CCA                   = 00000008
XWB$B_LOGIN                     = 000000CC              XWB$M_PRO_NAR                   = 00000010
XWB$B_LPRNAM                    = 000000A4              XWB$M_PRO_NFC                   = 00000001
XWB$B_PRO                       = 0000005A              XWB$M_PRO_PH2                   = 00000004
XWB$B_RID                       = 0000006F              XWB$M_PRO_SFC                   = 00000002
XWB$B_RPRNAM                    = 000000B8              XWB$M_STS_ASTPND                = 00000400
XWB$B_SP3                       = 0000006E              XWB$M_STS_ASTREQ                = 00000800
XWB$B_STA                       = 0000001E              XWB$M_STS_CON                   = 00000010
XWB$B_TYPE                      = 0000000A              XWB$M_STS_DIS                   = 00000008
XWB$B_X_FLW                     = 0000006C              XWB$M_STS_DTNAK                 = 00000100
XWB$B_X_FLWCNT                  = 0000006D              XWB$M_STS_LINAK                 = 00000200
XWB$C_COMLNG                    = 000000A4              XWB$M_STS_NDC                   = 00001000
XWB$C_CONLNG                    = 00000112              XWB$M_STS_OVF                   = 00000080
XWB$C_DATA                      = 00000010              XWB$M_STS_RBP                   = 00000040
XWB$C_LOGIN                     = 00000040              XWB$M_STS_SOL                   = 00000004
XWB$C_LPRNAM                    = 00000014              XWB$M_STS_TID                   = 00000001
XWB$C_NDC_LNG                   = 00000020              XWB$M_STS_TLI                   = 00000002
XWB$C_NUMSTA                    = 00000008              XWB$M_STS_TMO                   = 00000020
XWB$C_RID                       = 00000010              XWB$Q_FORK                      = 00000014
XWB$C_RPRNAM                    = 00000014              XWB$Q_FREE_CXB                  = 00000118
XWB$C_STA_CAR                   = 00000002              XWB$R_CON_BLK                   = 000000A4
XWB$C_STA_CCS                   = 00000004              XWB$R_RUN_BLK                   = 000000A4
XWB$C_STA_CIR                   = 00000003              XWB$S_                          = 00000006
XWB$C_STA_CIS                   = 00000001              XWB$S_COMLNG                    = 0000006E
XWB$C_STA_CLO                   = 00000000              XWB$S_CON_BLK                   = 0000006E
XWB$C_STA_DIR                   = 00000006              XWB$S_DATA                      = 00000010
XWB$C_STA_DIS                   = 00000007              XWB$S_DT                        = 00000030
XWB$C_STA_RUN                   = 00000005              XWB$S_FLG                       = 00000002
XWB$L_DEA_IRP                   = 00000104              XWB$S_FORK                      = 00000008
XWB$L_FPC                       = 00000020              XWB$S_FREE_CXB                  = 00000008
XWB$L_FR3                       = 00000024              XWB$S_LI                        = 00000030
XWB$L_FR4                       = 00000028              XWB$S_LOGIN                     = 0000003F
XWB$L_ICB                       = 0000010C              XWB$S_LPRNAM                    = 00000013
XWB$L_IRP_ACC                   = 00000080              XWB$S_NDC                       = 00000020
```

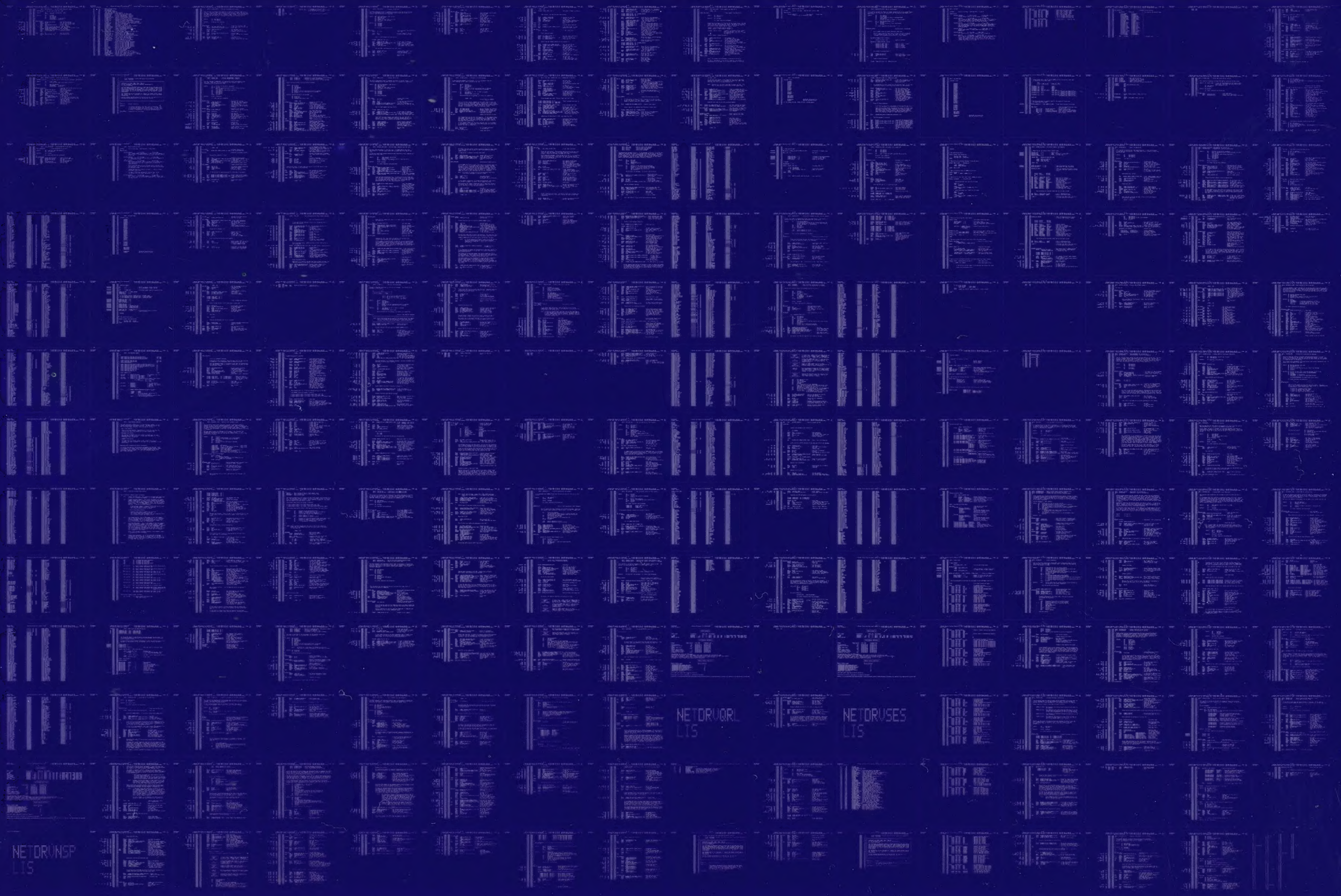
```
NET
V04
```

```
XWB$S_PRO                        = 00000001              XWB$W_PROGRESS                   = 00000052
XWB$S_RID                        = 00000010              XWB$W_REFCNT                     = 0000000C
XWB$S_RPRNAM                     = 00000013              XWB$W_REMLNK                     = 0000003C
XWB$S_RUN_BLK                    = 00000064              XWB$W_REMNOD                     = 0000003A
XWB$S_STS                        = 00000002              XWB$W_REMSIZ                     = 00000042
XWB$S_XWB                        = 00000120              XWB$W_RETRAN                     = 00000054
XWB$T                            = 00000112              XWB$W_R_REASON                   = 00000044
XWB$T_DATA                       = 0000005C              XWB$W_SIZE                       = 00000008
XWB$T_DT                         = 000000A4              XWB$W_STS                        = 0000000E
XWB$T_LI                         = 000000D4              XWB$W_TIMER                      = 00000050
XWB$T_LOGIN                      = 000000CD              XWB$W_TIM_ID                     = 00000048
XWB$T_LPRNAM                     = 000000A5              XWB$W_TIM_INACT                  = 0000004C
XWB$T_RID                        = 00000070              XWB$W_X_REASON                   = 00000046
XWB$T_RPRNAM                     = 000000B9              XWB$Z_NDC                        = 00000084
XWB$T_TR3HDR                     = 00000158  G           XWB_C_LEN                        = 0000017C
XWB$V_FLG_BREAK                  = 00000000              XWB_LOCLNK                         00000C97 R      03
XWB$V_FLG_CLO                    = 00000009              _$ACT_DFLT                       = 00000000
XWB$V_FLG_IAVL                   = 0000000C              _$ACT_INDEX                      = 00000016
XWB$V_FLG_SCD                    = 00000008              _$EVENT_INDEX                    = 00000015
XWB$V_FLG_SDACK                  = 00000003              _$MSK                            = 00007DFF
XWB$V_FLG_SDFL                   = 0000000E              _$TMP                            = 00000120  G
XWB$V_FLG_SDT                    = 00000007
XWB$V_FLG_SIACK                  = 00000002
XWB$V_FLG_SIFL                   = 0000000D
XWB$V_FLG_SLI                    = 00000004
XWB$V_FLG_TBPR                   = 0000000B
XWB$V_FLG_WBP                    = 00000006
XWB$V_FLG_WBUF                   = 00000001
XWB$V_FLG_WDAT                   = 0000000A
XWB$V_FLG_WHGL                   = 00000005
XWB$V_PRO_CCA                    = 00000003
XWB$V_PRO_NAR                    = 00000004
XWB$V_PRO_NFC                    = 00000000
XWB$V_PRO_PH2                    = 00000002
XWB$V_PRO_SFC                    = 00000001
XWB$V_STS_ASTPND                 = 0000000A
XWB$V_STS_ASTREQ                 = 0000000B
XWB$V_STS_CON                    = 00000004
XWB$V_STS_DIS                    = 00000003
XWB$V_STS_DTNAK                  = 00000008
XWB$V_STS_LINAK                  = 00000009
XWB$V_STS_NDC                    = 0000000C
XWB$V_STS_OVF                    = 00000007
XWB$V_STS_RBP                    = 00000006
XWB$V_STS_SOL                    = 00000002
XWB$V_STS_TID                    = 00000000
XWB$V_STS_TLI                    = 00000001
XWB$V_STS_TMO                    = 00000005
XWB$W_CI_PATH                    = 00000110
XWB$W_DELAY                      = 0000004E
XWB$W_DLY_FACT                   = 00000056
XWB$W_DLY_WGHT                   = 00000058
XWB$W_ELAPSE                     = 0000004A
XWB$W_FLG                        = 0000001C
XWB$W_LOCLNK                     = 0000003E
XWB$W_LOCSIZ                     = 00000040
XWB$W_PATH                       = 00000038
```

```
                                                   +----------------+
                                                   ! Psect synopsis !
                                                   +----------------+

PSECT name                        Allocation          PSECT No.  Attributes
----------                        ----------          ---------  ----------
.  ABS  .                         00000000  (     0.)  00 (   0.)  NOPIC    USR    CON    ABS    LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                             00000057  (    87.)  01 (   1.)  NOPIC    USR    CON    ABS    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
$$$105_PROLOGUE                   0000008E  (   142.)  02 (   2.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
$$$115_DRIVER                     00000D93  (  3475.)  03 (   3.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD     WRT NOVEC LONG

                                                +------------------------+
                                                ! Performance indicators !
                                                +------------------------+

Phase                       Page faults    CPU Time       Elapsed Time
-----                       -----------    --------       ------------
Initialization                       26    00:00:00.09    00:00:00.71
Command processing                  157    00:00:01.15    00:00:04.92
Pass 1                              990    00:00:45.45    00:01:28.11
Symbol table sort                     0    00:00:05.56    00:00:12.19
Pass 2                              520    00:00:10.73    00:00:19.59
Symbol table output                   0    00:00:00.67    00:00:00.95
Psect synopsis output                 2    00:00:00.05    00:00:00.15
Cross-reference output                0    00:00:00.00    00:00:00.00
Assembler run totals               1698    00:01:03.71    00:02:06.71

The working set limit was 2000 pages.
243296 bytes (476 pages) of virtual memory were used to buffer the intermediate code.
There were 180 pages of symbol table space allocated to hold 3212 non-local and 320 local symbols.
3029 source lines were read in Pass 1, producing 35 object records in Pass 2.
91 pages of virtual memory were used to define 70 macros.

                                               +----------------------------+
                                               ! Macro library statistics !
                                               +----------------------------+

Macro library name                             Macros defined
------------------                             --------------
_$255$DUA28:[SHRLIB]NMALIBRY.MLB;1                          0
_$255$DUA28:[SHRLIB]EVCDEF.MLB;1                            0
_$255$DUA28:[NETACP.OBJ]NETDRV.MLB;1                        3
_$255$DUA28:[NETACP.OBJ]NET.MLB;1                          10
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                             30
_$255$DUA28:[SYSLIB]STARLET.MLB;2                          10
TOTALS (all libraries)                                     53

3485 GETS were required to define 53 macros.

There were no errors, warnings or information messages.

MACRO/L:S=LIS$:NETDRVSES/OBJ=OBJ$:NETDRVSES MSRC$:NETDRVSES/UPDATE=(ENH$:NETDRVSES)+EXECML$/LIB+LIB$:NET/LIB+LIB$:NETDRV/LIB+SHRLIB$
```

0277 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0278 AH-BT13A-SE DIGITAL EQUIPMENT CORPORATION
VAX/VMS V4.0 CONFIDENTIAL AND PROPRIETARY